"Modnum"
Scilab toolbox
for the communication systems
User’s Guide

Draft Version

IRCOM Group
Alan Layec

February 8, 2006

Contents

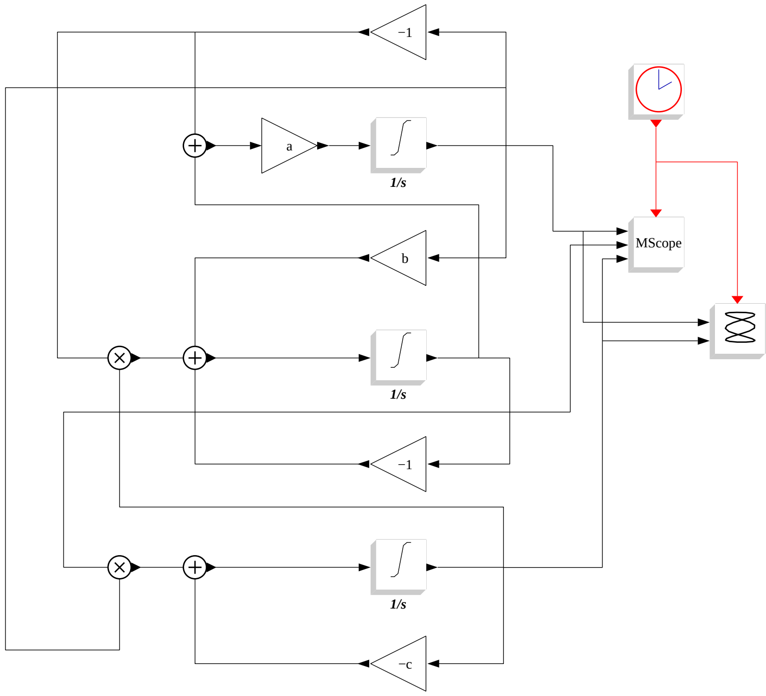
Part I

Scicos Diagrams

Chapter 1

Chaotic dynamical systems

1.1 Lorentz's system



1.1.1 Description

The Lorentz's system is defined by the following continuous system :

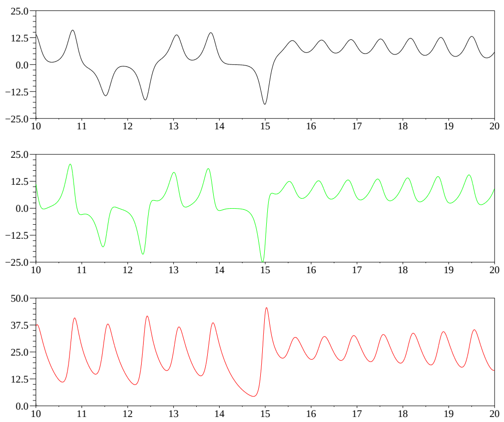
$$\frac{dx(t)}{dt} = a\left(-x(t) + y(t)\right) \quad (1.1)$$
$$\frac{dy(t)}{dt} = bx(t) - y(t) - x(t)y(t) \quad (1.2)$$
$$\frac{dz(t)}{dt} = -cx(t) + x(t)y(t) \quad (1.3)$$

The state variables $x(t)$, $y(t)$ and $z(t)$ are temperature of the air, speed of wind and a third caracteristic which represents the variation of temperature in accord to the altitude.

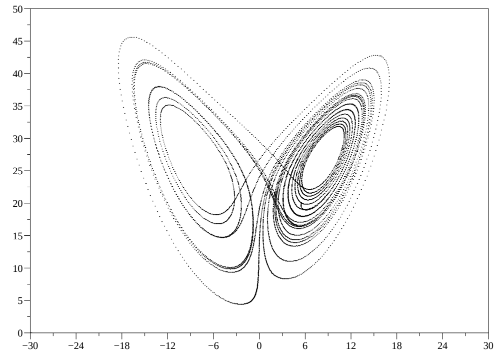
1.1.2 Context

```
Tsampl=3e-3
a=10
b=28
c=8/3
ci=[5.5;5;20]
Tfin=20
```

1.1.3 Scope Results

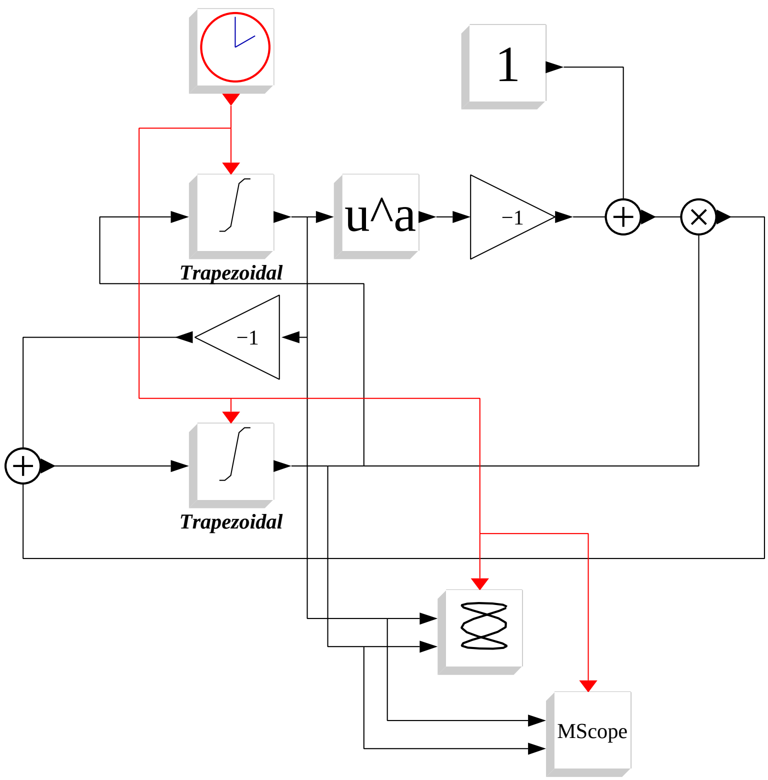


(a) Temporal wave forms of state variables



(b) Phase plan

1.2 Discrete Van Der Pol's system (trapezoidal méthode)



1.2.1 Description

The Van Der Pol's Equation is written with the equation :

$$x + \left(x^2 - 1\right)\dot{x} + x = 0 \tag{1.4}$$

It is fully described by the following system of state equation :

$$\dot{x} = y \tag{1.5}$$

$$\dot{y} = \left(1 - x^2\right) y - x \tag{1.6}$$

1.2.2 Context

```
Te=0.01
Tfin=90
ci=[-1;0]
```

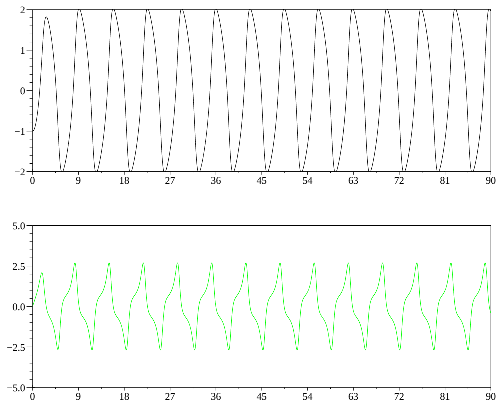
1.2.3 Scope Results

1.2.4 Mod_num blocks

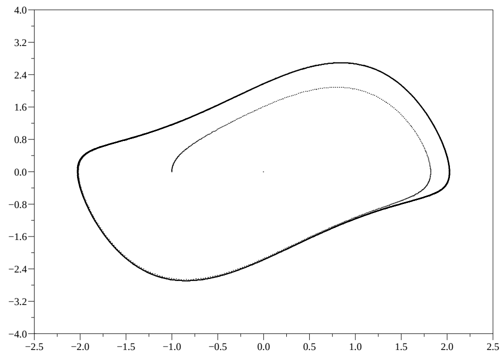
- TRAPINTEGRAL_f - Single step integrator block by trapezoidal method

1.2.5 See Also

- EULERINTEGRAL_f - Single step integrator block by Euler method (Scicos Block)

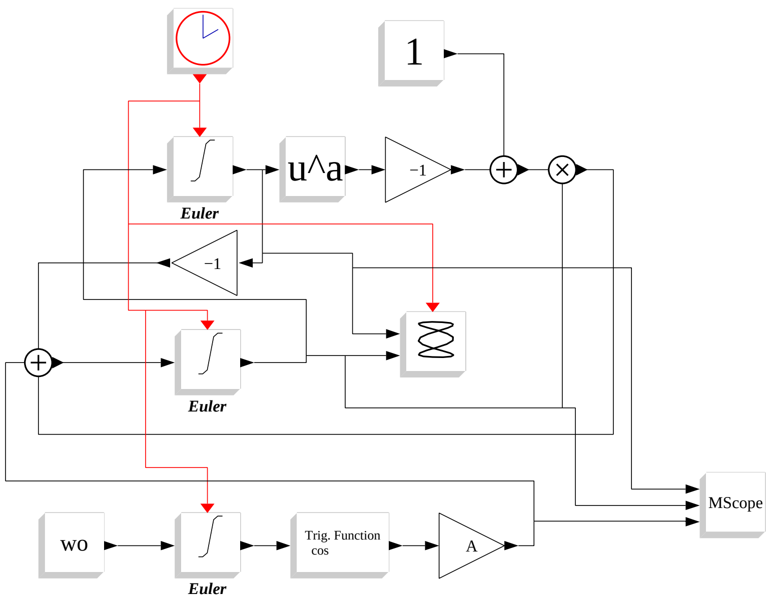


(a) Time domain wave forms of state variables



(b) Phase plan

1.3 Discrete forced Van Der Pol's system (Euler method)



1.3.1 Description

This system illustrates the use of a single step integrator in order to resolve continuous system. Here the system is resolved with the backward Euler method. The forced Van Der Pol's oscillator is described with the system of state equation :

$$\dot{x} = y \tag{1.7}$$

$$\dot{y} = \left(1 - x^2\right) y - x + A\cos\left(z\right) \tag{1.8}$$

$$\dot{z} = \frac{2\pi}{T} \tag{1.9}$$

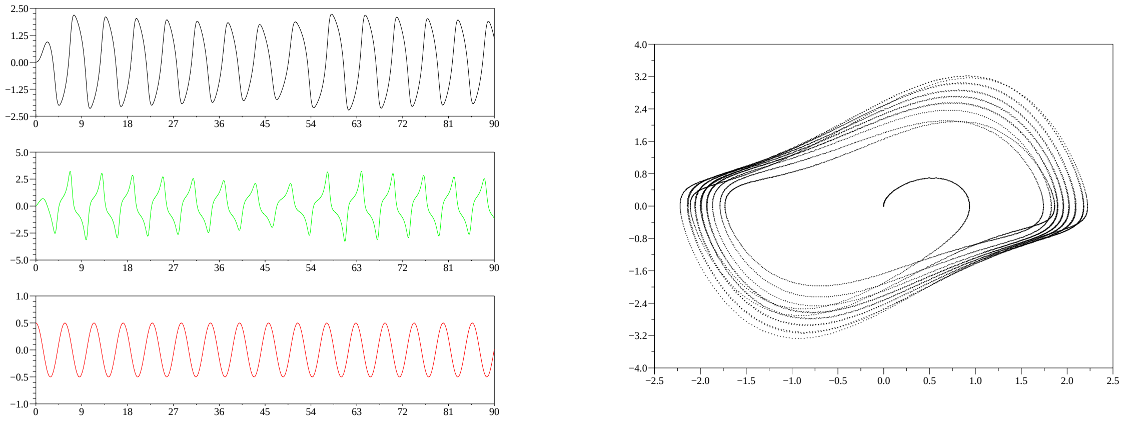
where parameters T is the period of the driven oscillator and A it's amplitude.

1.3.2 Context

```
Te=0.01
```

```
Tfin=90
ci=[-1;0]
A=0.5;
wo=1.1;
To=2*%pi/wo;
```

1.3.3 Scope Results



(a) Time domain wave forms of state variables

(b) Phase plan

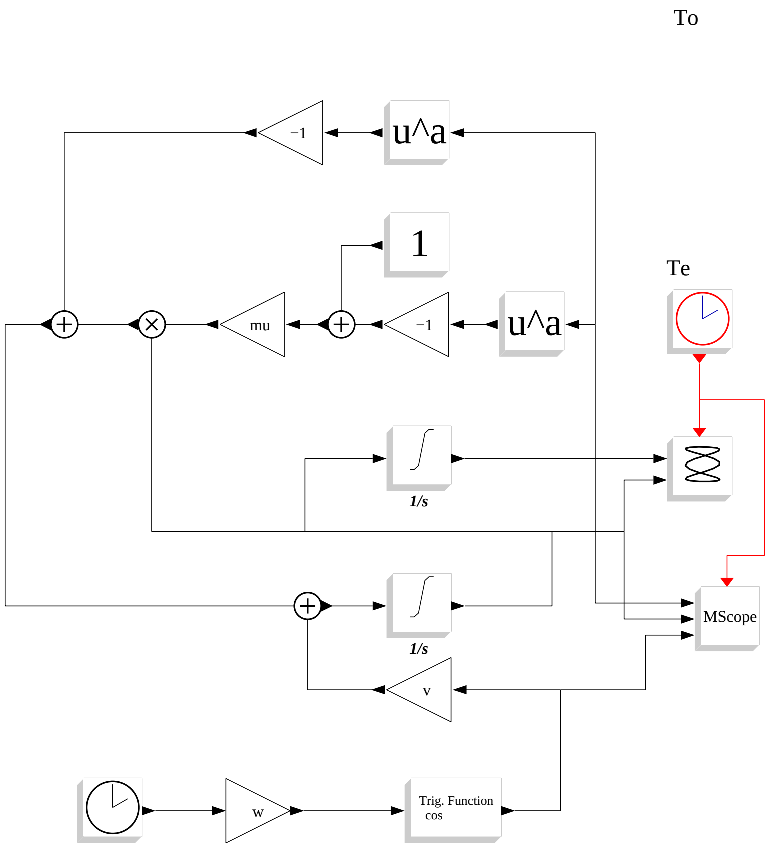
1.3.4 Mod_num blocks

- EULERINTEGRAL_f - Single step integrator block by Euler method

1.3.5 See Also

- EULERINTEGRAL_f - Single step integrator block by Euler method (Scicos Block)

1.4 Duffing-Van Der Pol's system



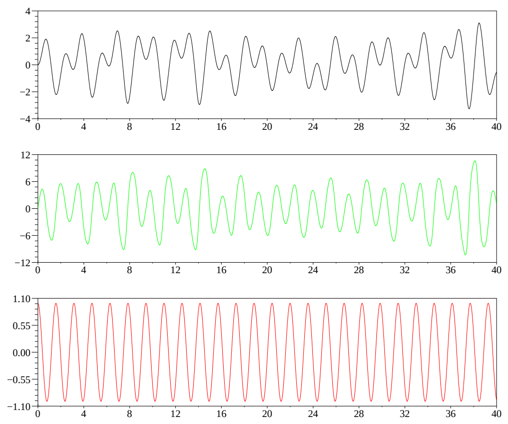
1.4.1 Description

Add here a paragraph of the function description.

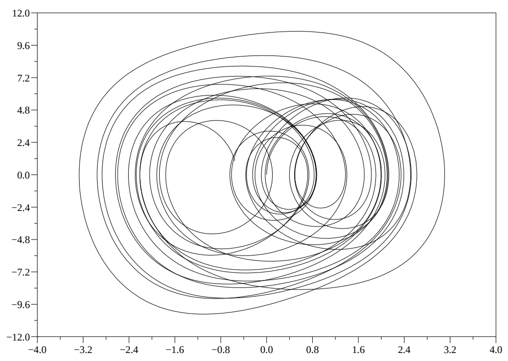
1.4.2 Context

```
Te = 5e-3
mu=0.2
v=17
w=4
To=2*%pi/w
Tfin = 40
```

1.4.3 Scope Results

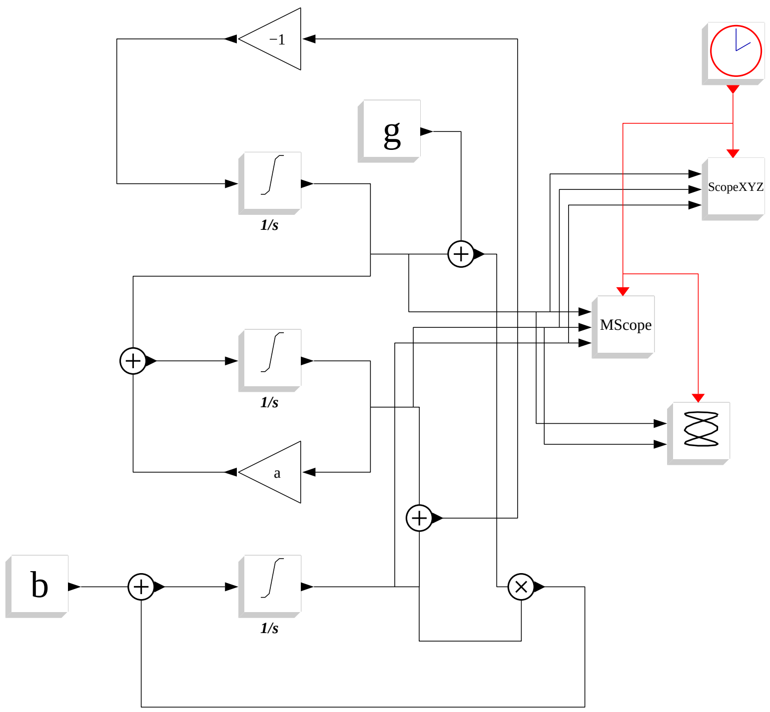


Scope results



Scope results

1.5 Rössler's system



1.5.1 Description

The Rössler's system is defined with the following system of state equations :

$$\tilde{X} = -(Y+Z) \tag{1.10}$$

$$\tilde{Y} = X + \alpha Y \tag{1.11}$$

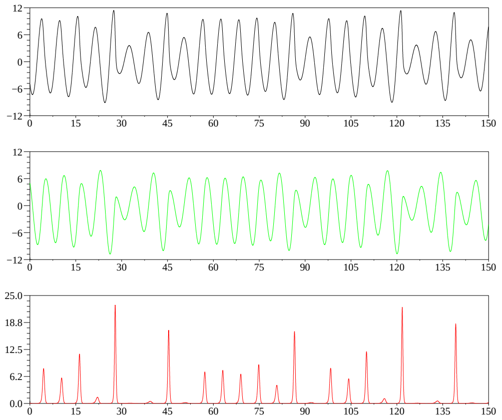
$$\tilde{Z} = \beta + Z(X-\gamma) \tag{1.12}$$

1.5.2 Context

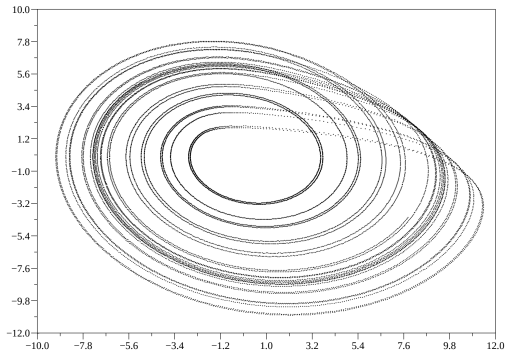
```
Te=0.01
```

```
a=0.2
b=0.2
g=-5.7
ci=[-5;5;0.1]
Tfin = 150
```

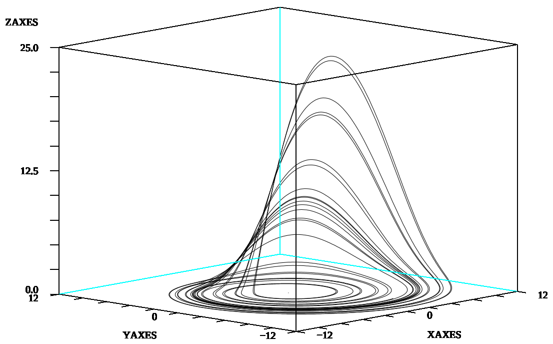
1.5.3 Scope Results



(a) Time domain wave forms of state variables



(b) Phase plan

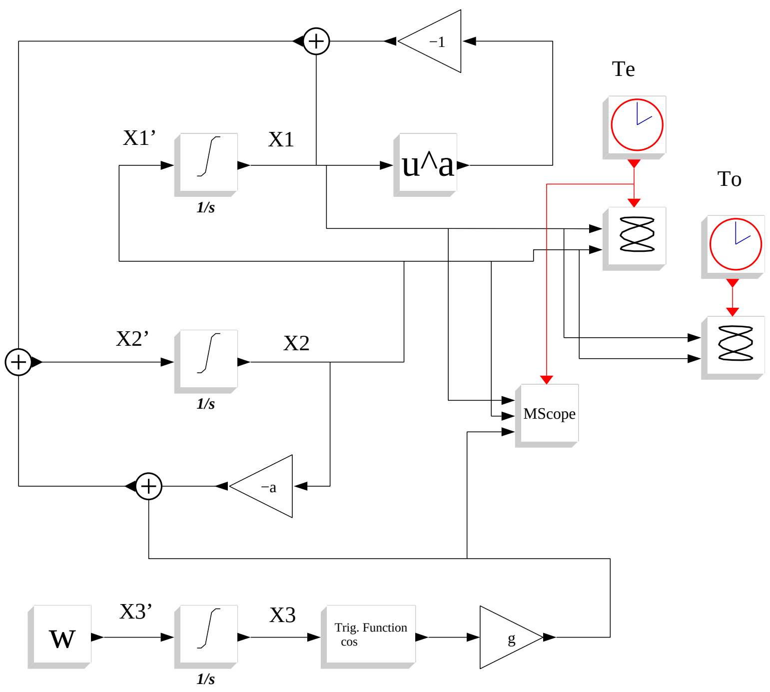


(c) 3d phase plan

1.5.4 Mod_num blocks

- SCOPXYZ_f - 3D trajectory scope block

1.6 Duffing's oscillator



1.6.1 Description

The Duffing's equation is described by this non-linear differential equation :

$$\frac{y\,(t)}{dt} = x\,(t) - x^3\,(t) - \epsilon y\,(t) + \gamma \cos\,(\omega t) \tag{1.13}$$

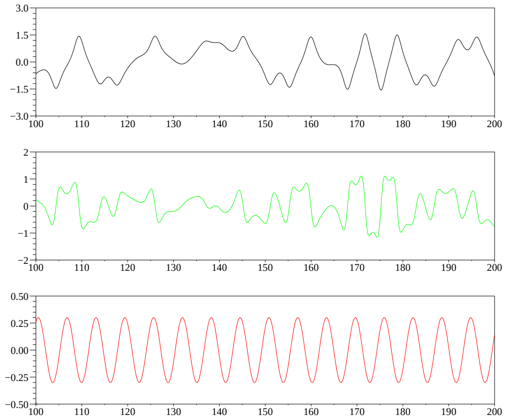
where ϵ and γ are parameters and ω a pulsation.
This forced oscillator should be written as a three dimensional state equation system :

$$\tilde{x_1} = x_2 \tag{1.14}$$
$$\tilde{x_2} = x_1 - x_1^3 - \epsilon y + \gamma \cos\,(x_3) \tag{1.15}$$
$$\tilde{x_3} = \frac{2\pi}{T} \tag{1.16}$$

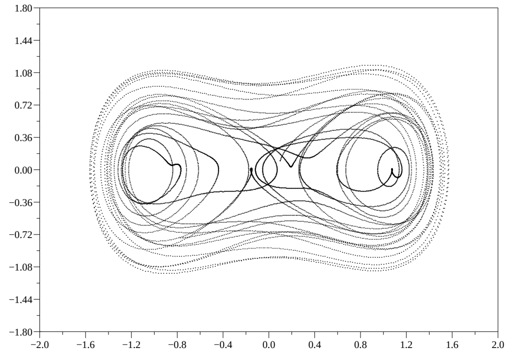
1.6.2 Context

```
Te=0.02
Tfin=200
g=0.3
a=0.15
w=1
To=2*%pi/w
ci1=0.1
ci2=0.1
ci3=0
```

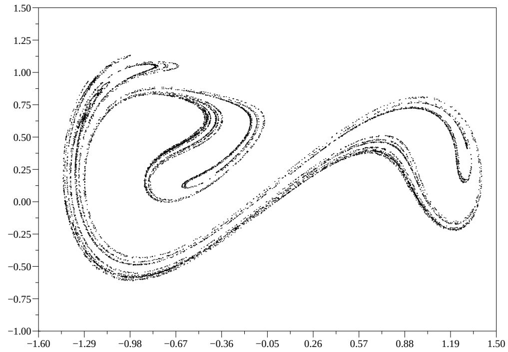
1.6.3 Scope Results



(a) Time domain wave forms

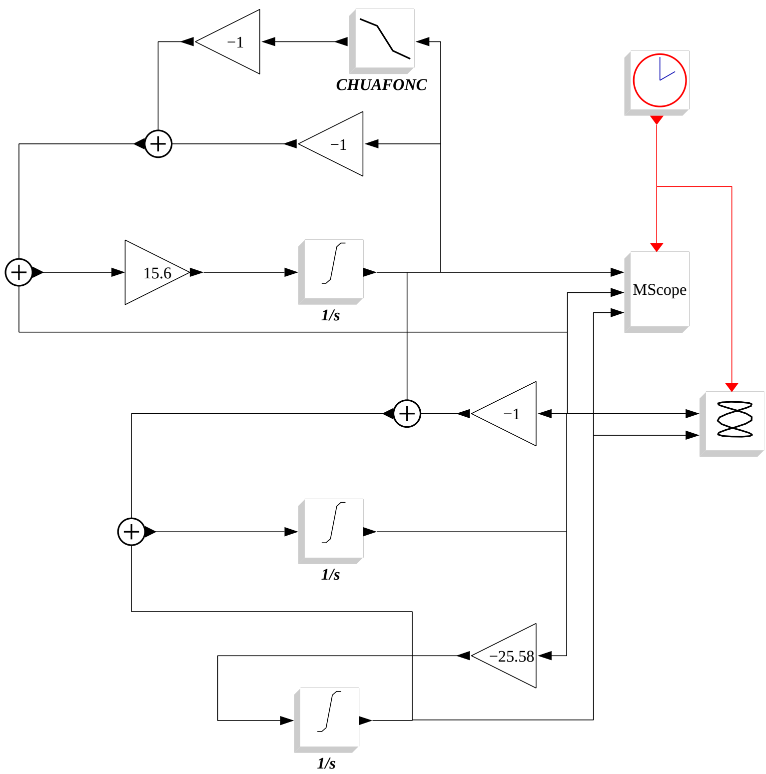


(b) Phase plan



(c) Poincare section

1.7 Chua's system



1.7.1 Description

This system takes an important place in the understanding of the synchronization of chaotic systems. He is originally studed with the electronic circuit shown in the following Figure. This circuit is composed with passive elements and with an active non-linear element.

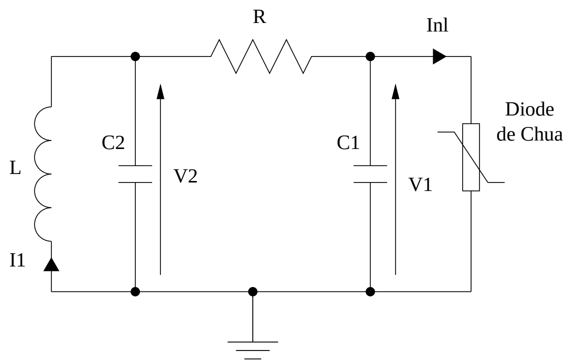


Figure 1.1: Chua's circuit

This circuit is described by the following state equations:

$$\begin{aligned} \frac{dV_1}{dt} &= \frac{1}{C_1}\left[\frac{1}{R}\left(V_2 - V_1\right) - i_{NL}\right] \\ \frac{dV_2}{dt} &= \frac{1}{C_2}\left[i_l - \frac{1}{R}\left(V_2 - V_1\right)\right] \\ \frac{di_l}{dt} &= -\frac{1}{L}V_2 \end{aligned} \tag{1.17}$$

$$f(V_1) = G_b\, V_1 + \frac{1}{2}\left[G_a - G_b\right]\left[|V_1 + E| - |V_1 - E|\right] \tag{1.18}$$

By changing parameters with $\alpha = \frac{C_2}{C_1}$, $\beta = \frac{C_2}{L}R^2$, $m_0 = \frac{G_a}{R}$, $m_1 = \frac{G_b}{R}$and renamming state variables, the new sate equations are:

$$\begin{aligned} \tilde{x}_1 &= \alpha\left[x_2 - x_1 - h\left(x_1\right)\right] \\ \tilde{x}_2 &= x_3 - x_2 - x_1 \\ \tilde{x}_3 &= -\beta x_2 \end{aligned} \tag{1.19}$$

$$h(x) = m_1\, x + \frac{1}{2}\left[m_0 - m_1\right]\left[|x + E| - |x - E|\right] \tag{1.20}$$

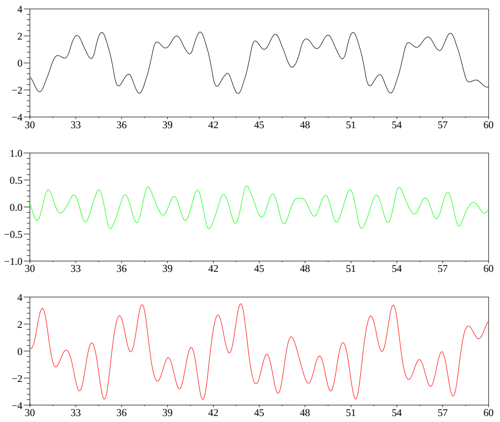
1.7.2 Context

```
Tsampl = 1e-2
Tfin = 60
m0=-8/7
m1=-5/7
ci=[1.6;0;-1.6]
```

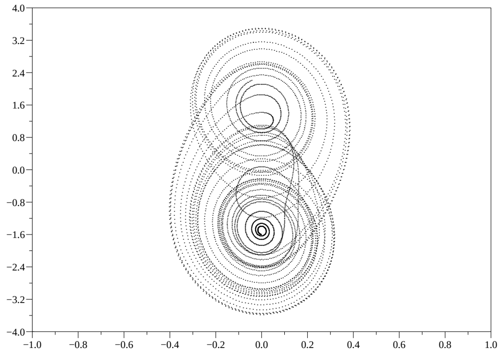
1.7.3 Scope Results

1.7.4 Mod_num blocks

- CHUAFONC_f - Chua's non-linear function block



(a) Temporal wave forms of state variables

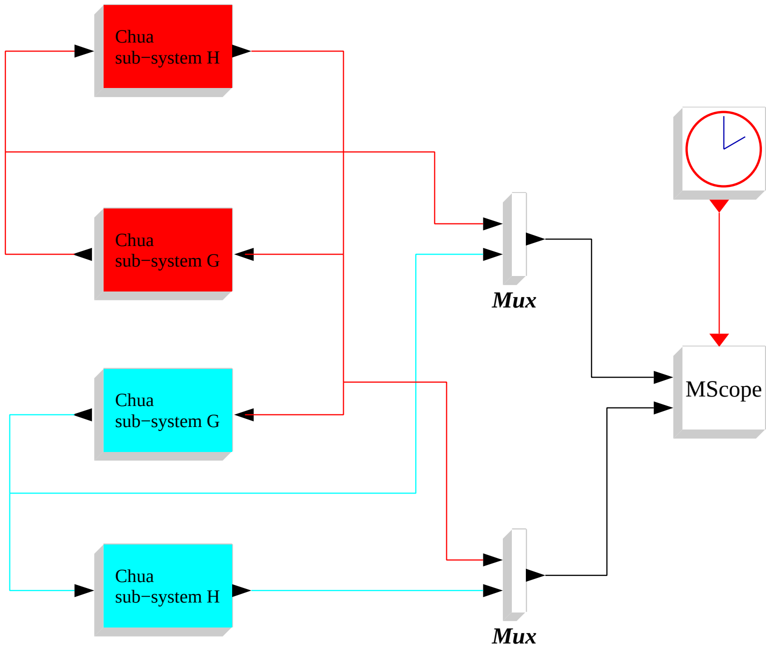


(b) Phase plan

1.7.5 See Also

- SYSTEMG_f - Chua sub-system G block (Scicos Block)
- SYSTEMH_f - Chua sub-system H block (Scicos Block)

1.8 Sub-system of Chua's circuit



1.8.1 Description

The Chua's circuit, shown in Fig.1.2 is able to be decomposed in two sub-systems G et H.

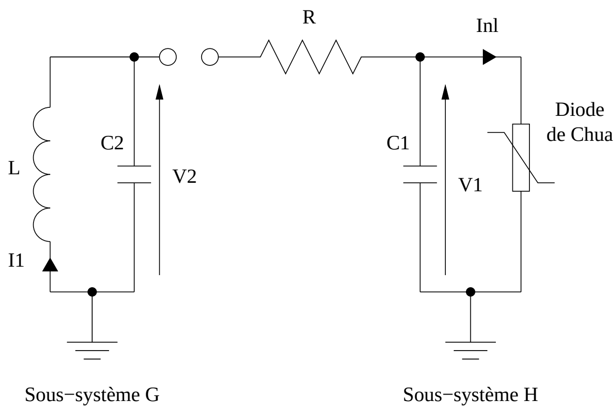


Figure 1.2: Decomposition of Chua's circuit

Making this decomposition, the Chua's circuit should be presented by an closed loop system (see Fig.1.3).

1.8.2 Context

```
Tsampl = 1e-2
Tfin = 30
m0=-8/7
m1=-5/7
ci_master=[1.6;0;-1.6]
ci_slave=[0;0;0]
```

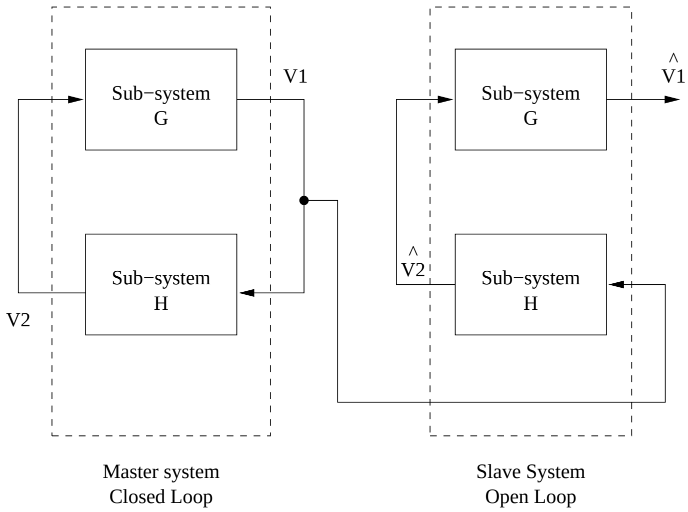
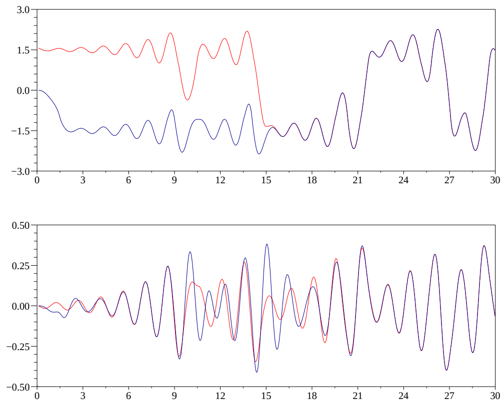


Figure 1.3: System of Chua's circuit



Temporal wave forms of state variables : (red line) master system; (blue line) slave system

1.8.3 Scope Results

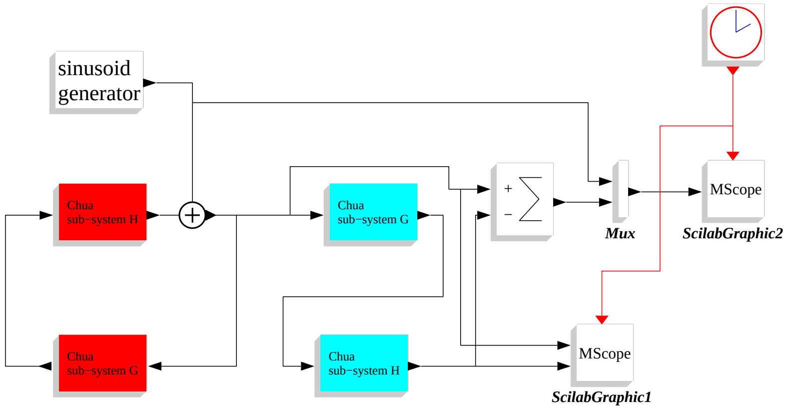
1.8.4 Mod_num blocks

- SYSTEMG_f - Chua sub-system G block
- SYSTEMH_f - Chua sub-system H block

1.8.5 See Also

- SYSTEMG_f - Chua sub-system G block (Scicos Block)
- SYSTEMH_f - Chua sub-system H block (Scicos Block)

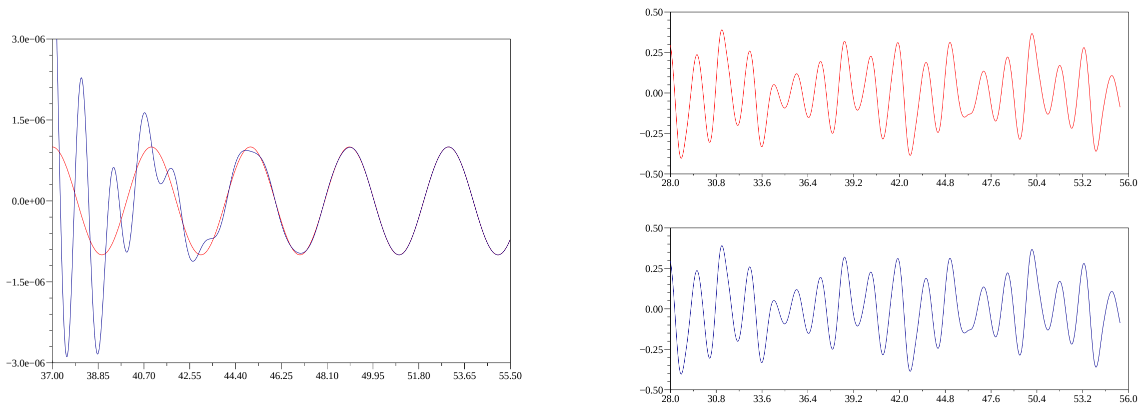
1.9 Master-slave synchronization of Chua's circuit



1.9.1 Context

```
Tsampl = 10e-3
Tfin = 55.5
m0=-8/7
m1=-5/7
ci_master=[1.6;0;-1.6]
ci_slave=[0;0;0]
ampl=1e-6
T=4
```

1.9.2 Scope Results



(a) red line: master system output wave form; blue line: slave system output wave form

(b) red line: master system input message; blue line: slave system output message

1.9.3 Mod_num blocks

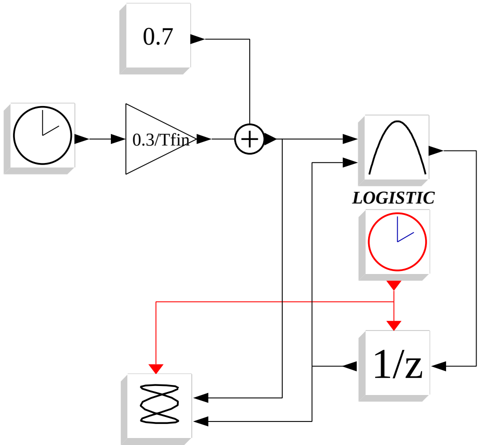
- SYSTEMG_f - Chua sub-system G block
- SYSTEMH_f - Chua sub-system H block

1.9.4 See Also

- SYSTEMG_f - Chua sub-system G block (Scicos Block)

- SYSTEMH_f - Chua sub-system H block (Scicos Block)

1.10 2D bifurcation of the logistic map



1.10.1 Description

The logistic equation is defined by the one-dimensional discrete-time system given by :

$$x[k+1] = f(x[k], R) = 4Rx[k](1 - x[k]) \tag{1.21}$$

This application should be modelled by the following block diagram Fig.1.4, where output of the non-linear function F(t) is feed-backed to the input via a delayed block.

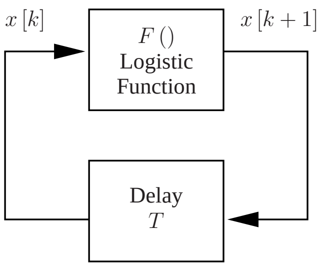


Figure 1.4: Block diagram of logistic application

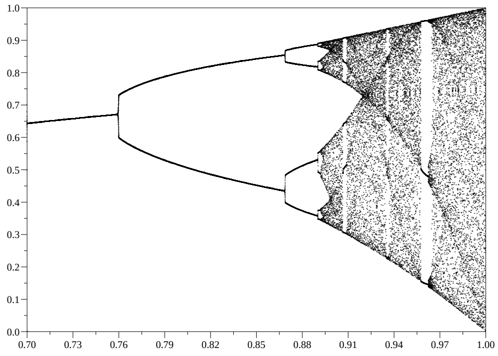
1.10.2 Context

```
Te = 1
ci=0.5
Tfin = 0.5e5*Te
```

1.10.3 Scope Results

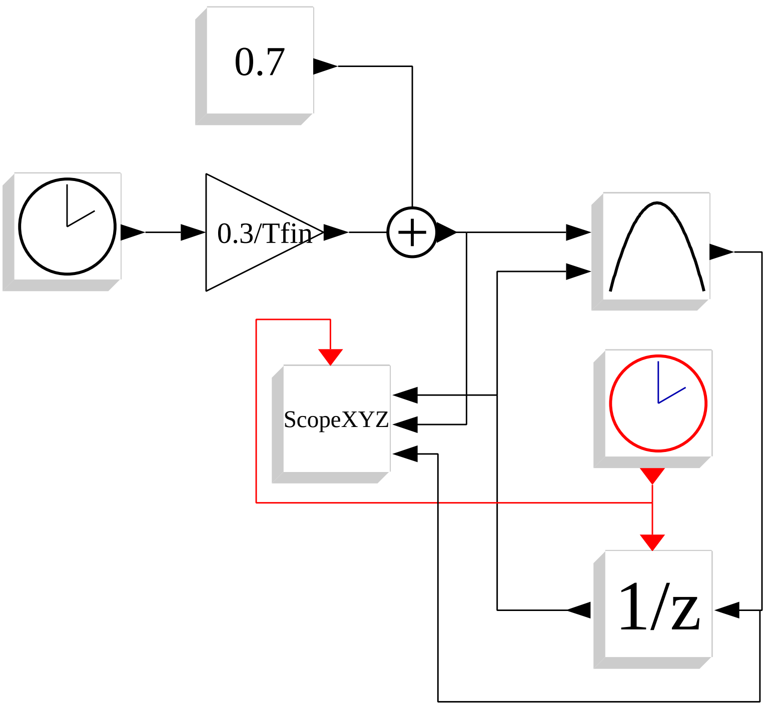
1.10.4 Mod_num blocks

- LOGISTIQUE_f - Logistic function block



Bifurcation diagram

1.11 3D bifurcation of the logistic map



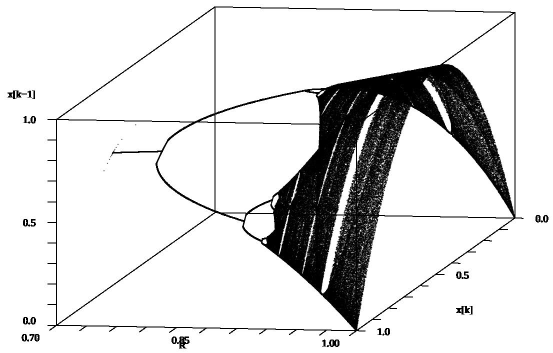
1.11.1 Description

Add here a paragraph of the function description.

1.11.2 Context

```
Te = 1
ci=0.5
Tfin = 5e5*Te
```

1.11.3 Scope Results

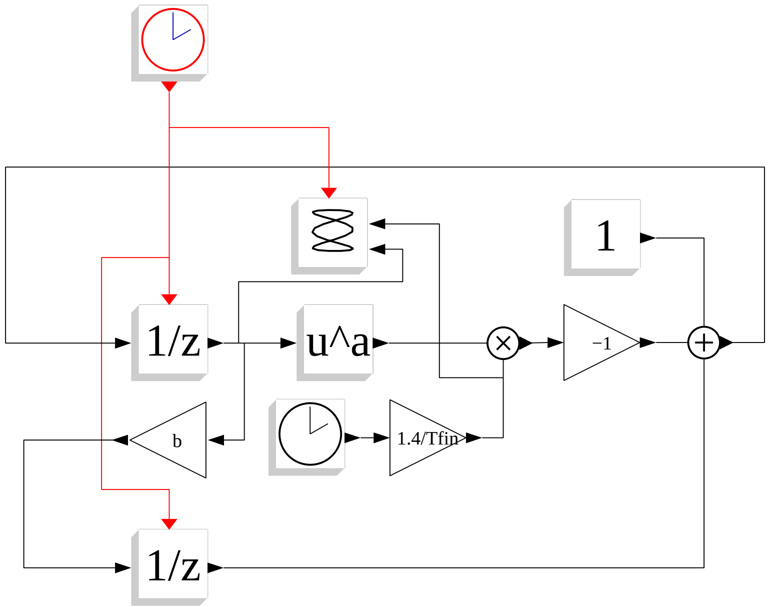


Scope results

1.11.4 Mod_num blocks

- LOGISTIQUE_f - Logistic function block
- SCOPXYZ_f - 3D trajectory scope block

1.12 Henon's system



1.12.1 Description

Henon's map is defined with the followind discret state sytem :

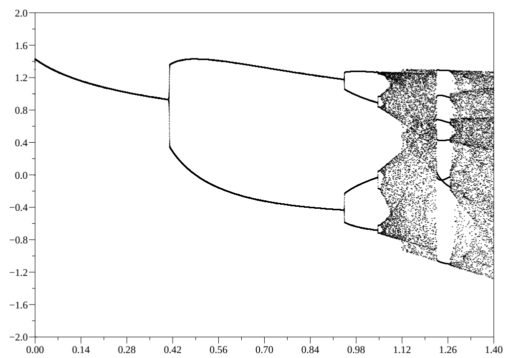
$$x_{k+1} = y_k + 1 - \alpha x_k^2 \tag{1.22}$$

$$y_{k+1} = \beta x_k \tag{1.23}$$

1.12.2 Context

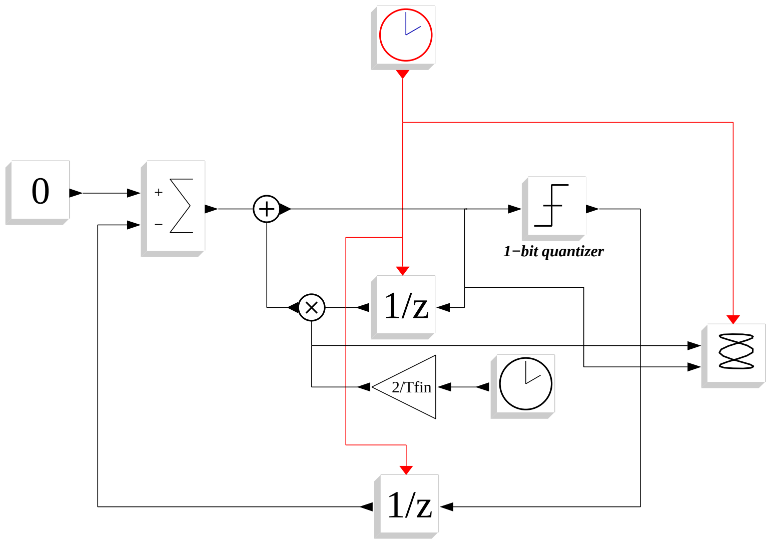
```
Te=1
Tfin=5e4*Te
b=0.3
```

1.12.3 Scope Results



Bifurcation diagram

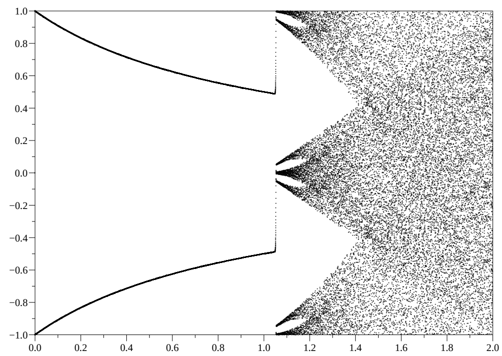
1.13 Chaotic first order Delta-Sigma modulator



1.13.1 Context

```
Te=1
Tfin=5e4*Te
```

1.13.2 Scope Results

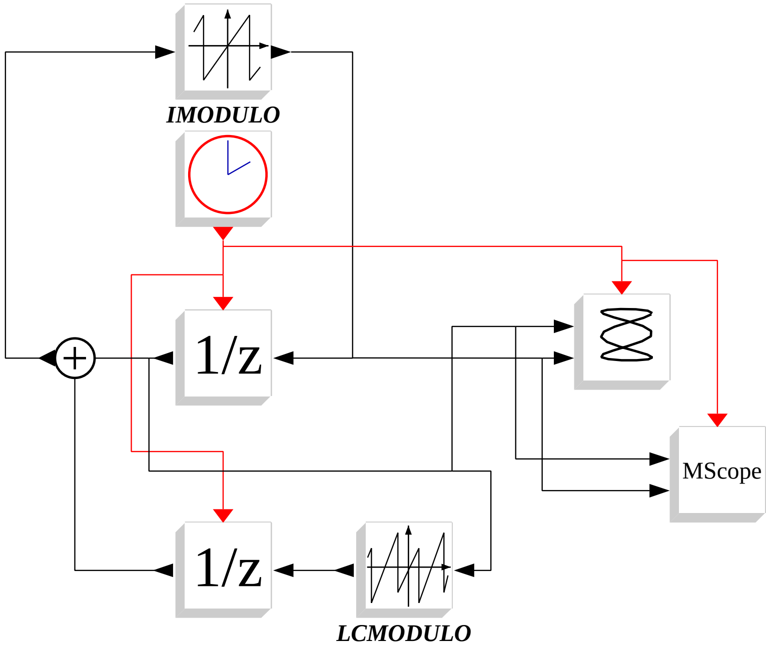


Bifurcation diagram

1.13.3 Mod_num blocks

- COMP_f - One bit Quantizer block

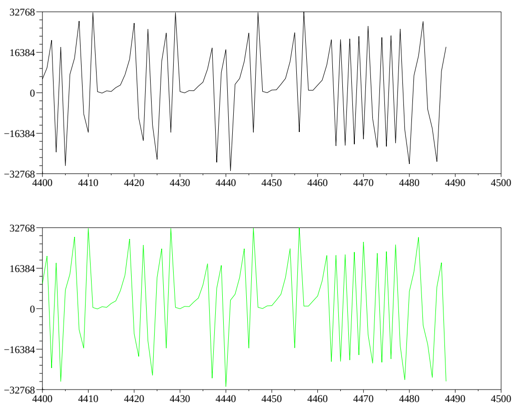
1.14 Frey's chaotic encoder



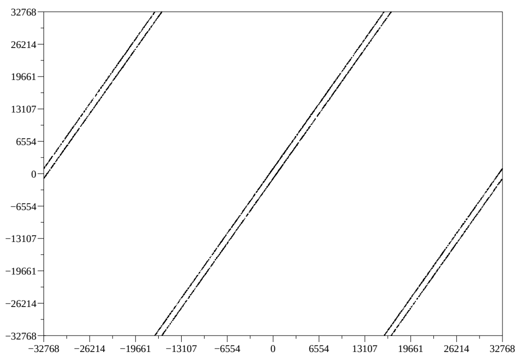
1.14.1 Context

```
Te = 1
Tfin = 4490
Nbit=16
Intmax = 2^(Nbit-1)
N = 1
ci1 = 1000
ci2 = 0
```

1.14.2 Scope Results



(a) time domain wave forms of state variables



(b) phase plan

1.14.3 Mod_num blocks

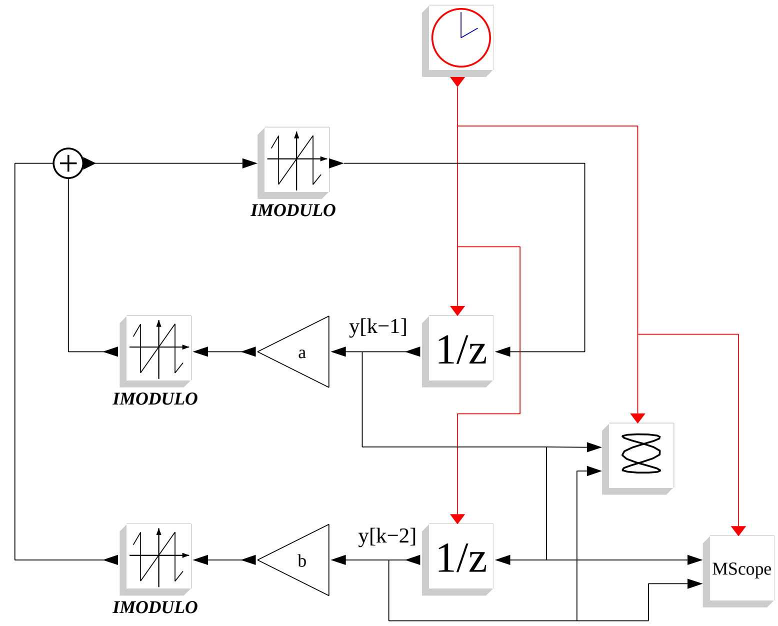
- IMODULO_f - Integer modulo function block

- LCMODULO_f - Left shift circulate integer modulo function block

1.14.4 See Also

- IMODULO_f - Integer modulo function block (Scicos Block)
- IMODULOB_f - Integer modulob function block (Scicos Block)
- FMODULOB_f - Float modulob function block (Scicos Block)
- FMODULOC_f - Float moduloc function block (Scicos Block)
- LCMODULO_f - Left shift circulate integer modulo function block (Scicos Block)

1.15 Second order Infinite Impulse Response filter



1.15.1 Description

The Fig. 1.5 presents the block diagram of an ideal second order digital filter.

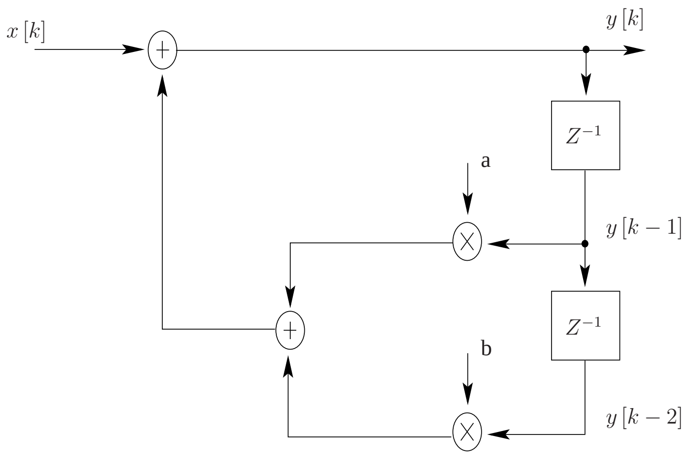


Figure 1.5: Block diagram of a linear second order digital recursive filter

The differential discret equation that described the system is given by:

$$y(z) = y(z)\left(az^{-1} + bz^{-2}\right) + x(z) \tag{1.24}$$

This linear system is traditionaly studed by the Z function transfert :

$$H(z) = \frac{y(z)}{x(z)} = \frac{1}{1 + az^{-1} + bz^{-2}} \tag{1.25}$$

With $H(Z)$, we determinate the stability domain of the filter in accord to the two gain paramter a et b. The Fig.1.6 represents the stability domain of the second order recursive filter.

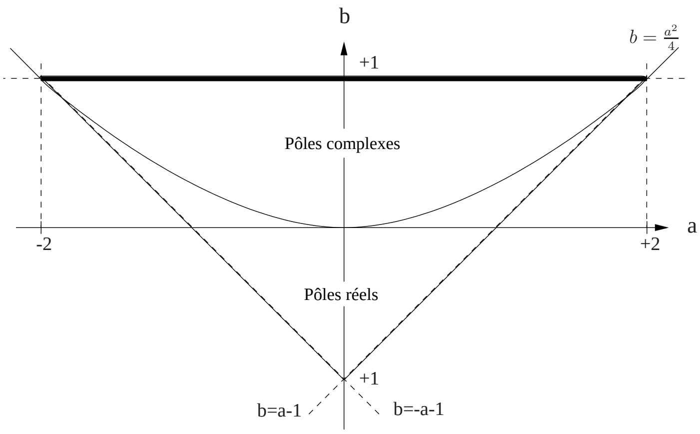


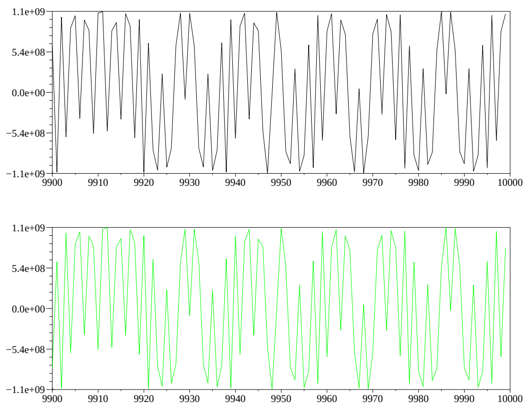
Figure 1.6: Stability domain of the second order recursive filter

Now, we replace the ideal operators (adder and multiplactor) by real operaters that are presents in Digital Signal Processors (DSP) when realising recursive filters. The associated non-linear function is the modulo function, wich represents the effect of the overflow that occurs in digital implementation. The non-linear system is then described by a non-linear discret state equations system :

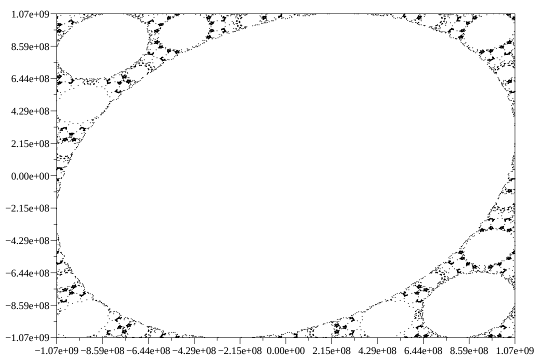
1.15.2 Context

```
Te = 1
Tfin = 1e4*Te
Nbit = 31
a=0.5
b=-1
ci=0.675
Intmax = 2^(Nbit-1);
ci1=-int(ci*Intmax)
ci2=int(ci*Intmax)
```

1.15.3 Scope Results



(a) Temporal wave forms of discrete state variables

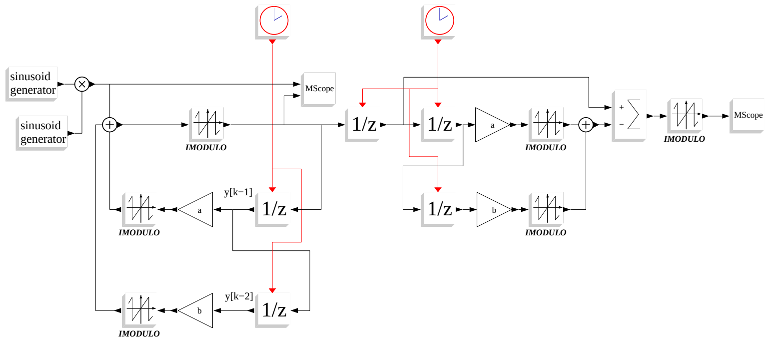


(b) Phase plan

1.15.4 Mod_num blocks

- IMODULO_f - Integer modulo function block

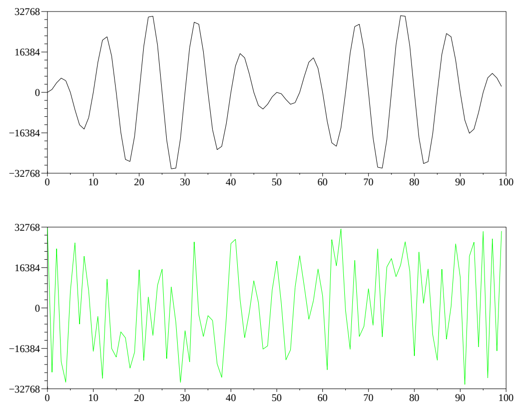
1.16 Master-slave synchronization of second order IIR filter



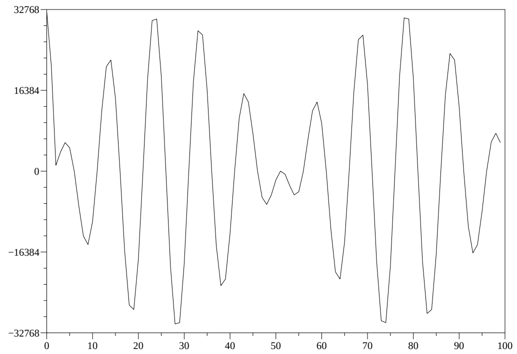
1.16.1 Context

```
Te = 1
Te_rec = 1
Tfin = 100
Nbit = 16
a=0.5
b=-1
ci=0.675
Intmax = 2^(Nbit-1);
ci1=-int(ci*Intmax)
ci2=int(ci*Intmax)
p_source=2*%pi*0.01
a_source=2^(Nbit-1)-1
```

1.16.2 Scope Results



(a) input message of emitter; output message of emitter



(b) output message of receiver

1.16.3 Mod_num blocks

- IMODULO_f - Integer modulo function block

1.16.4 See Also

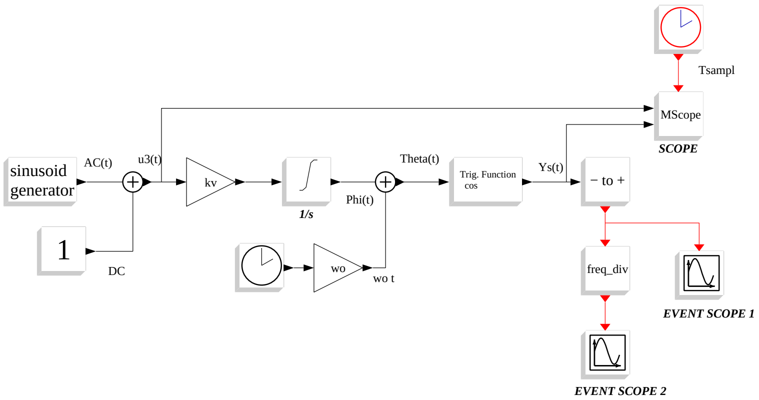
- IMODULO_f - Integer modulo function block (Scicos Block)
- IMODULOB_f - Integer modulob function block (Scicos Block)
- FMODULO_f - Float modulo function block (Scicos Block)

- FMODULOB_f - Float modulob function block (Scicos Block)
- FMODULOC_f - Float moduloc function block (Scicos Block)

Chapter 2

Oscillators and Phase Locked Loop systems

2.1 Continous Voltage Controled Oscillator



2.1.1 Description

This diagram realizes an open loop model of a continous Voltage Controled Oscillator.
Regular output of the VCO is described with the following formula :

$$y_s\left(t\right) = \cos\left[\left(\omega_0 t\right) + \int_0^t k_v u_3\left(\tau\right) d\tau\right] \tag{2.1}$$

Integration of input voltage and detection of crossing are made with the integral block and zeros crossing block of scicos which use the integrated lsoda solver.

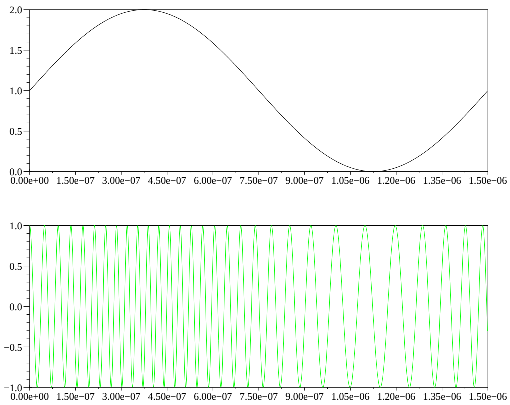
2.1.2 Context

```
Fo = 10e6;
To = 1 / Fo;
wo=2*%pi * Fo;
alpha=60e6;
kv=60e6
j_vco=100e6;
N=3;
Nsampl = 80;
Tsampl = To/(Nsampl);
Fsampl = 1 / Tsampl;
Tfin=15*To
```

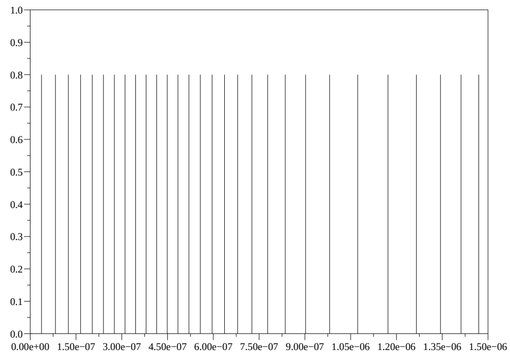
2.1.3 Scope Results

2.1.4 See Also

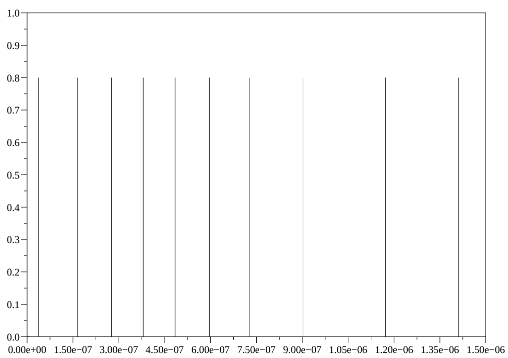
- discr_vco - Discrete Voltage Controled Oscillator (Scicos Diagram)



Input and output voltage of VCO



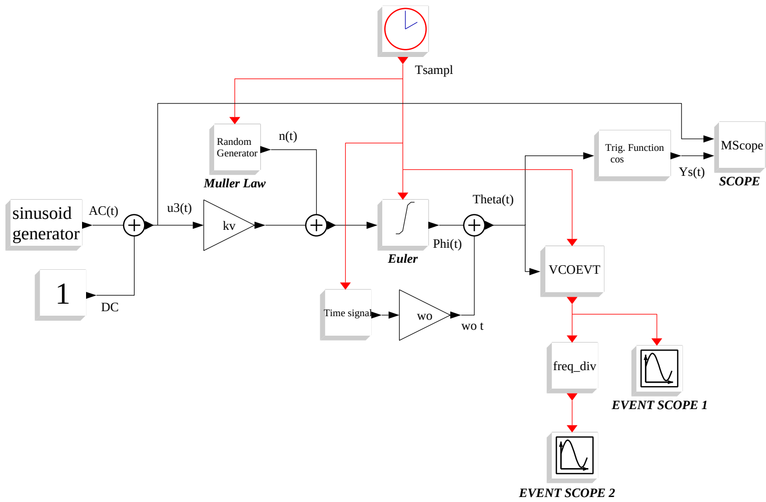
Events generated by the - to + block



Events generated by the frequency divider block

- VCO_f **-** Discrete Voltage Controlled Oscillator block (Scicos Block)

2.2 Discrete Voltage Controled Oscillator



2.2.1 Description

This diagram is an open loop model of a discrete Voltage Controled Oscillator.
Regular output voltage can be described with this continous equation :

$$y_s\,(t) = \cos\left[(\omega_0 t) + \int_0^t k_v u_3\,(\tau) + n(\tau)d\tau\right] \tag{2.2}$$

The noise introduces in the the tunning voltage ist a White Gaussian Noise.
The integral of the input voltage is realized with the EULERINTEGRAL block which use the discrete backward-Euler formula and the VCOEVT block produces the event corresponding to a zero crossing with a discrete method of detection.

2.2.2 Context

```
Fo = 10e6;
To = 1 / Fo;
wo=2*%pi * Fo;
alpha=60e6;
kv=60e6
j_vco=100e6;
N=3;
Nsampl = 80;
Tsampl = To/(Nsampl);
Fsampl = 1 / Tsampl;
Tfin=15*To
```

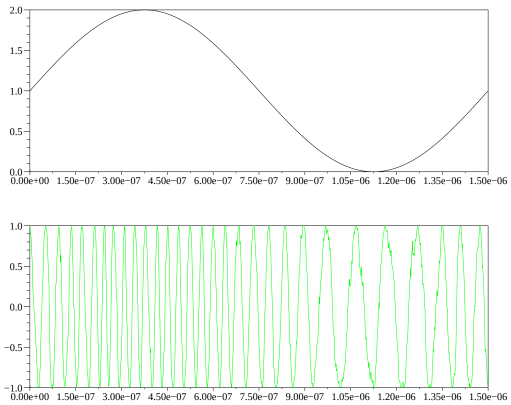
2.2.3 Scope Results

2.2.4 Mod_num blocks

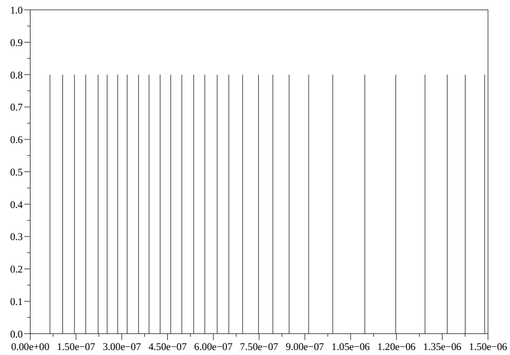
- NOISEBLK_f - Gaussian White Noise Generator block
- Time_f - Discrete time estimator bloc
- VCOEVT_f - Discrete zero crossing block
- EULERINTEGRAL_f - Single step integrator block by Euler method

2.2.5 See Also

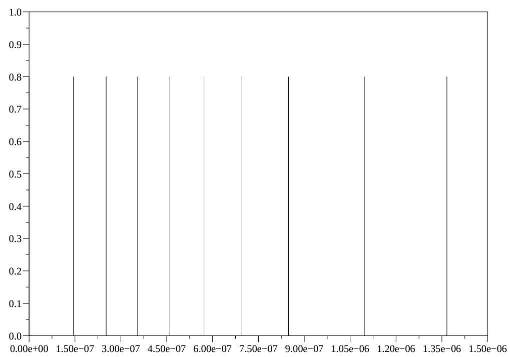
- scicos_vco - Continous Voltage Controled Oscillator (Scicos Diagram)



Input and output voltage of VCO

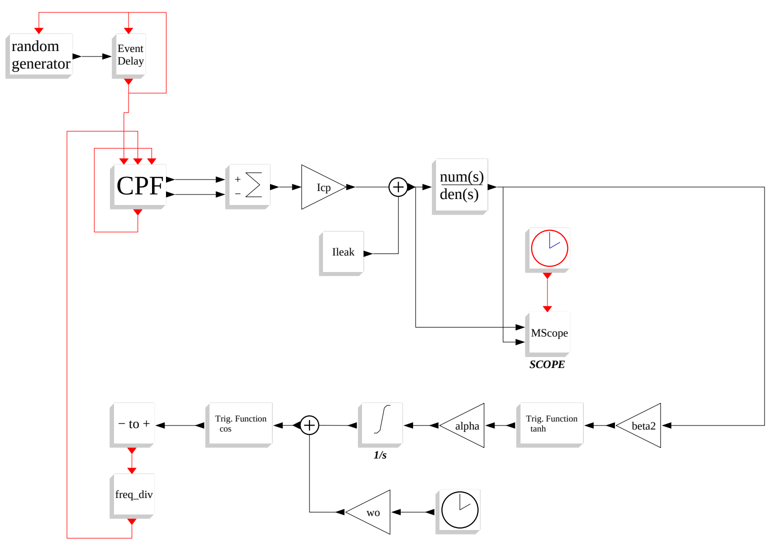


Events generated by the VCOEVT block



Events generated by the frequency divider block

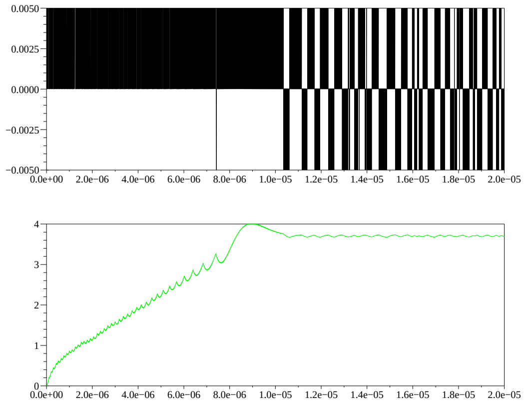
2.3 Simple scicos diagram of an integer-N frequency synthesizer



2.3.1 Context

```
F_cpf = 50e6;
T_cpf = 1/F_cpf;
sig_ref=T_cpf*0.1/100

Fo = 2.045e9;
To = 1 / Fo;
wo=2*%pi * Fo;
kv = 100.5e6;
alpha=6.91e9;
beta2=0.15;
j_vco=1e6;

N=52;
Fd=N*F_cpf;

Nsampl = 4;
Tsampl = 1/(Fd*Nsampl);
Fsampl = 1 / Tsampl;

Icp = 5e-3;
Ileak=10e-6;
fn=F_cpf/180;
phi=%pi/4;

kv=kv*2*%pi
[tau1,tau,tau2]=calcul_3eme_ordre(fn,phi,kv,Icp,N);
s=poly(0,'s');
num=1+tau1*s;
den=tau*s*(1+tau2*s);
kv=kv/(2*%pi)
Tfin=1000*T_cpf
```

2.3.2 Scope Results

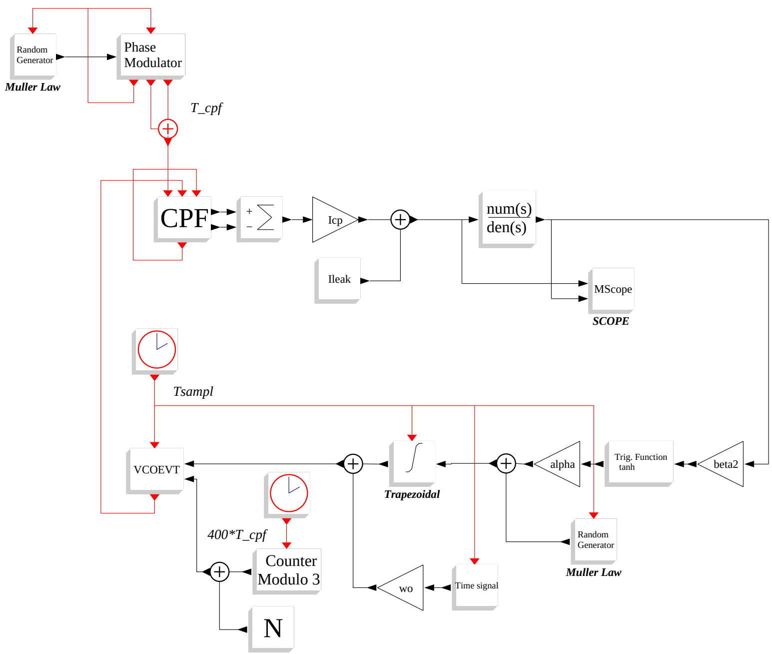


Scope results

2.3.3 Mod_num blocks

- CPF_f - Tri-state Phase/Frequency Comparator block

2.4 Scattered diagram of a third order integer-N frequency synthesizer



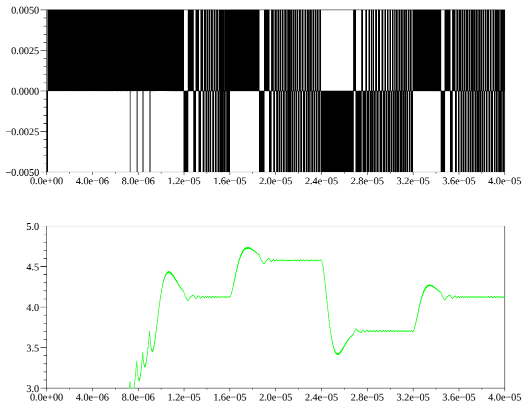
2.4.1 Context

```
lines(-1);
F_cpf = 50e6;
T_cpf = 1/F_cpf;
sig_ref=T_cpf*0.2/100

Fo = 2.045e9;
To = 1 / Fo;
wo=2*%pi * Fo;
kv = 100.5e6;
alpha=6.91e9;
beta2=0.15;
j_vco=1e6;

N=52;
Fd=N*F_cpf;

Nsampl = 2;
Tsampl = 1/(Fd*Nsampl);
Fsampl = 1 / Tsampl;

Icp = 5e-3;
Ileak=1e-6;
fn=F_cpf/180;
phi=%pi/4;

kv=kv*2*%pi
[tau1,tau,tau2]=calcul_3eme_ordre(fn,phi,kv,Icp,N);
s=poly(0,'s');
num=1+tau1*s;
den=tau*s*(1+tau2*s);
kv=kv/(2*%pi)
Tfin=2e3*T_cpf
```

2.4.2 Scope Results

2.4.3 Mod_num blocks

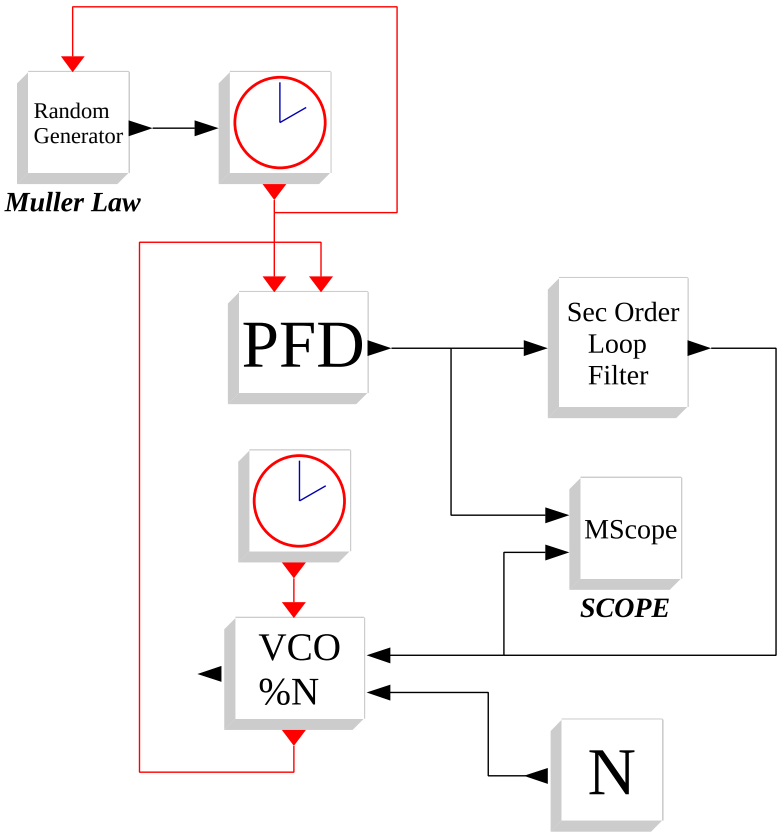
- TRAPINTEGRAL_f - Single step integrator block by trapezoidal method
- VCOEVT_f - Discrete zero crossing block
- CPF_f - Tri-state Phase/Frequency Comparator block
- Time_f - Discrete time estimator bloc



Scope results

- NOISEBLK_f - Gaussian White Noise Generator block
- PEVTDLY_f - Phase modulator event generator block

2.5 Integrated diagram of an integer-N frequency synthesizer



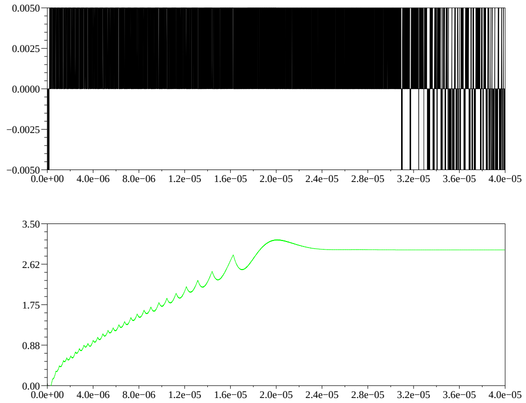
2.5.1 Context

```
F_cpf = 20e6;
T_cpf = 1/F_cpf;
sig_ref=(2*%pi*0.2/100)

Fo = 2.045e9;
To = 1 / Fo;
wo=2*%pi * Fo;
kv = 100.5e6;
alpha=6.91e9;
beta2=0.15;
j_vco=1e6;

Fd=2.5e9;
N=int(Fd/F_cpf)

Nsampl = 2;
Tsampl = T_cpf/(Nsampl);
Fsampl = 1 / Tsampl;

Icp = 5e-3;
Ileak=1e-6;
Ileak_sig=1e-12;
fn=F_cpf/180;
phi=%pi/4;

Tfin=0.8e3*T_cpf
```

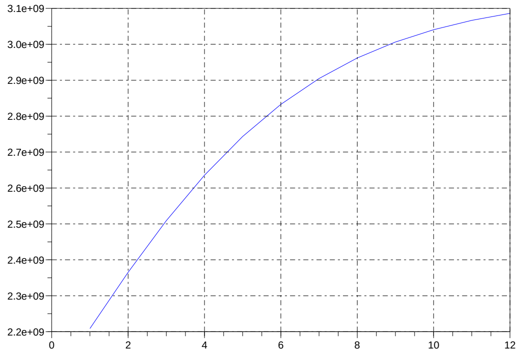
2.5.2 Scope Results

2.5.3 Mod_num blocks

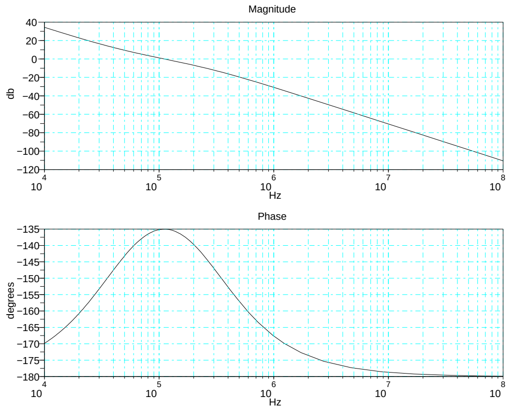
- PFD_f - Phase/Frequency tristate Detector block
- VCO_f - Discrete Voltage Controlled Oscillator block



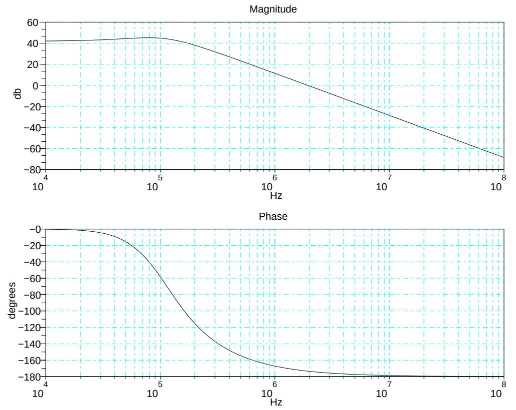
Time domain wave forms (a)Charge pump output current; (b)VCO input voltage



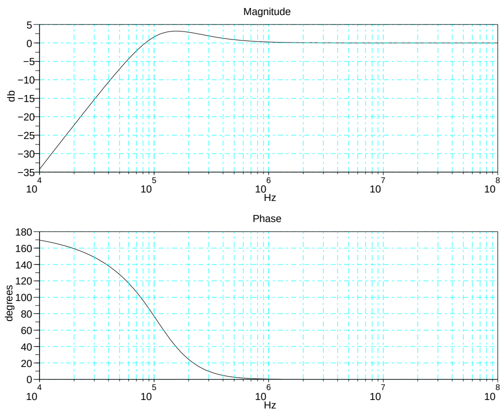
Non linear voltage/frequency caracteristic of VCO



Open-loop transfert function



Closed-loop transfert function

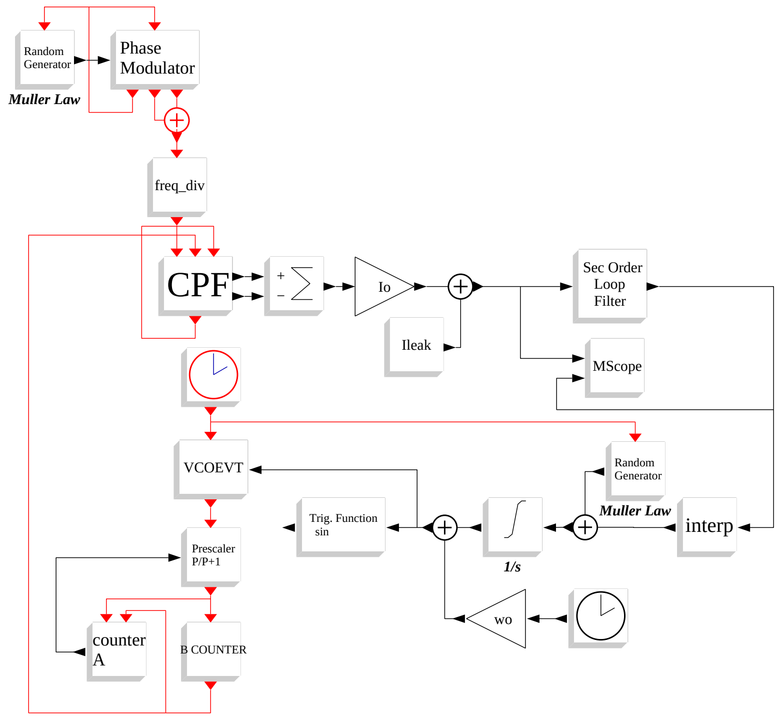


FM-response transfert function

- PCLOCK_f - Integrated phase modulator event generator block
- NOISEBLK_f - Gaussian White Noise Generator block

- SOLOOPFILTER_f - Second order Loop Filter for PLL type 2 block

2.6 Scattered diagram of a third order integer-N frequency synthesizer with an interpoled non-linearity of VCO and with dual-modulus feedback divider



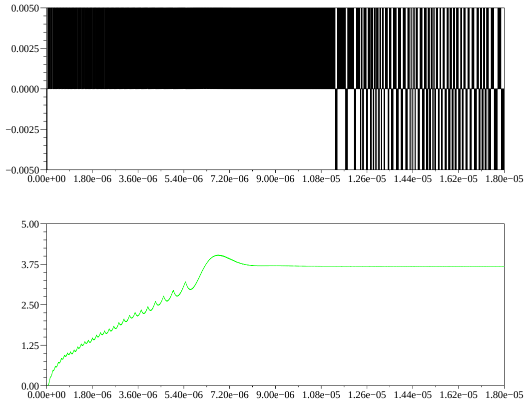
2.6.1 Context

```
F_ref=250e6
T_ref=1/F_ref
sig_ref=0.2*T_ref
R=5
F_cpf = F_ref/R;
T_cpf = 1/F_cpf;

Io=5e-3
Ileak=1e-8

Fo = 2.045e9;
To = 1 / Fo;
wo=2*%pi * Fo;
j_vco=2e6;

y_tab=read(MODNUM+'/scs_diagr/pll/synthe/vcotab',12,9);
d_tab=splin(y_tab(:,1),((y_tab(:,2)*1e6)-Fo)*2*%pi)
i_tab=1:0.1:12;
f_tab=interp(i_tab,y_tab(:,1),((y_tab(:,2)*1e6)-Fo)*2*%pi,d_tab);

A=4;
B=6;
P=8;
N=B*P+A;

Fd=N*F_cpf;;

fn=F_cpf/180
mphi=%pi/4
kv=80e6

Nsampl = 2;
Tsampl = 1/(Nsampl*Fd);
Fsampl = 1 / Tsampl;
Tfin=0.9e3*T_cpf
```

2.6.2 Scope Results

2.6.3 Mod_num blocks

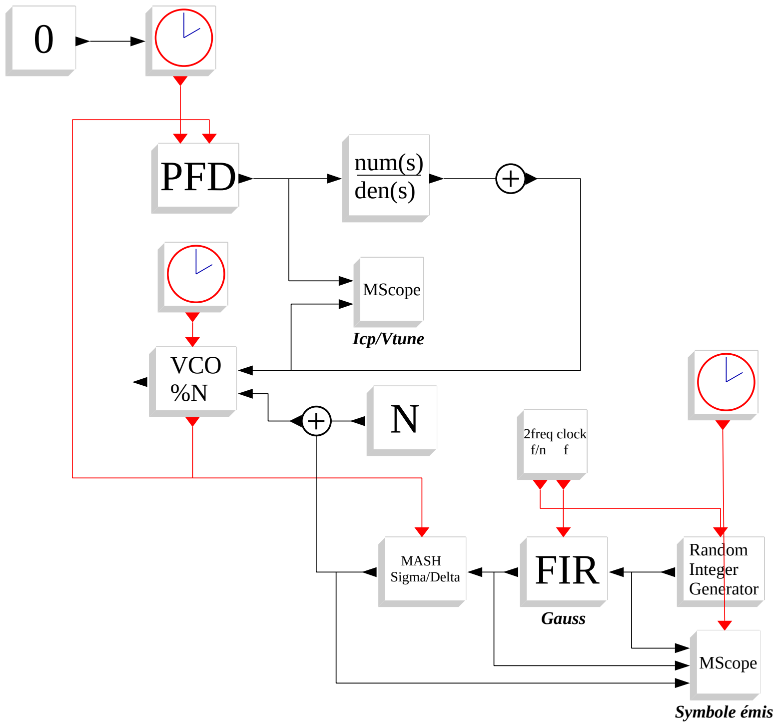
- NOISEBLK_f - Gaussian White Noise Generator block



Scope results

- PEVTDLY_f - Phase modulator event generator block
- SOLOOPFILTER_f - Second order Loop Filter for PLL type 2 block
- VCOEVT_f - Discrete zero crossing block
- CPF_f - Tri-state Phase/Frequency Comparator block

2.7 Modulated fractional frequency synthesizer



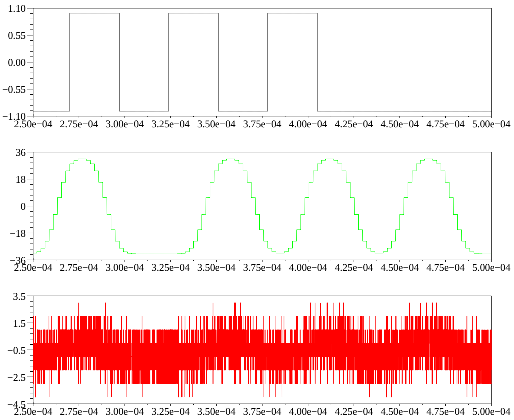
2.7.1 Context

```
lines(-1);
F_cpf = 20e6;
T_cpf = 1/F_cpf;
sig_ref=T_cpf*0.02/100

Fo = 2.045e9;
To = 1 / Fo;
wo=2*%pi * Fo;
kv = 100.5e6;
alpha=6.91e9;
beta2=0.15;
j_vco=0.2e6;

Fd=2.5e9;
N=int(Fd/F_cpf)

Nsampl = 2;
Tsampl = 1/(F_cpf*Nsampl);
Fsampl = 1 / Tsampl;

Icp = 5e-3;
Ileak=0.1e-6;
fn=F_cpf/180;
phi=%pi/4;

Ts=3/fn
Nech=12
Te=Ts/Nech
Nbit=3
M=32

[tau1,tau,tau2]=calcul_3eme_ordre(fn,phi,kv*2*%pi,Icp,N);
s=poly(0,'s');
num=1+tau1*s;
den=tau*s*(1+tau2*s);
Tfin=10e3*T_cpf
```

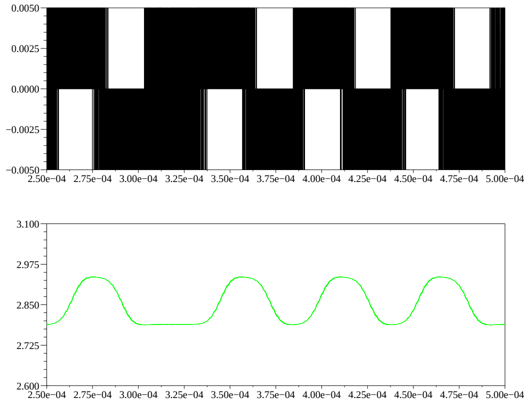
2.7.2 Scope Results

2.7.3 Mod_num blocks

- PFD_f - Phase/Frequency tristate Detector block



Scope results



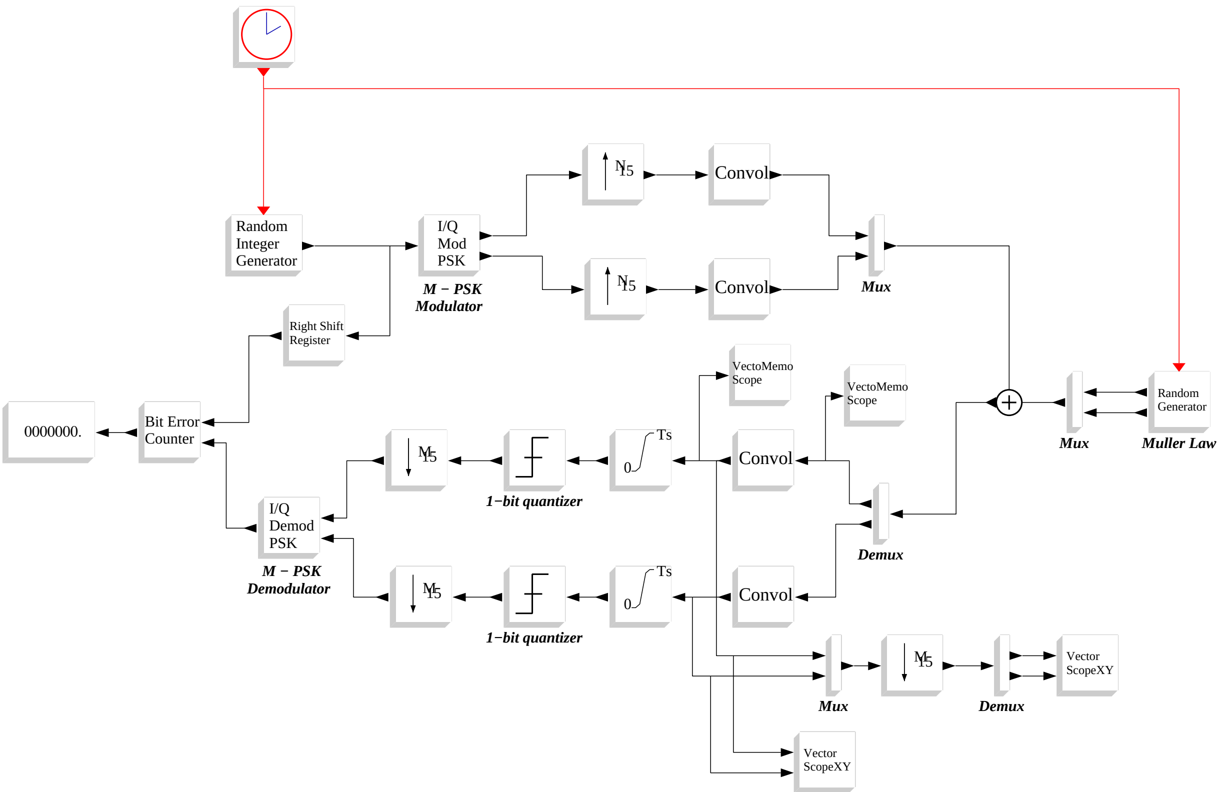
Scope results

- VCO_f - Discrete Voltage Controlled Oscillator block
- PCLOCK_f - Integrated phase modulator event generator block
- MASHBLK_f - Delta-Sigma modulator block
- RIFGEN_f - Generic Finite Impulse Response filter block
- GENINT_f - Random Integer Generator block

Chapter 3

Communication systems

3.1 Scattered diagram of base band QPSK transmission chain



3.1.1 Description

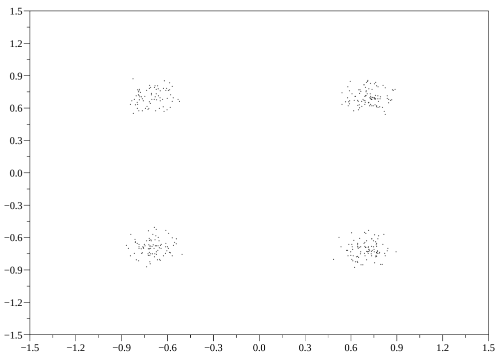
Long description

3.1.2 Context

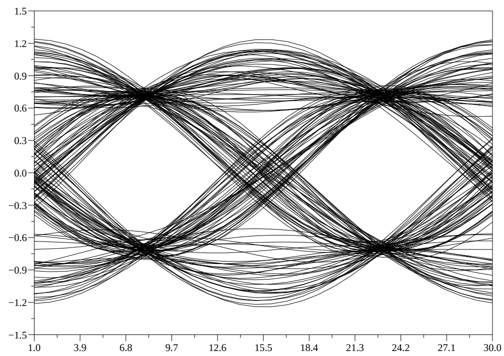
```
Te=0.1;
Nu=350;
Nech=15;
N=Nu*Nech;
Nbit=2;
nb_coef=127;
r=0.35;
```

```
fe=Nech;
gain_em=1;
gain_rec=1/Nech;
pulse=zeros(1,1);
t_Ts=1/fe*(-nb_coef/2:nb_coef/2-1);
pulse = 4*r/%pi*(cos((1+r)*%pi*t_Ts) + (sin((1-r)*%pi*t_Ts)./(4*r*t_Ts)))./(1-(4*r*t_Ts).^2);
Nb_vec=3;
nb_event=3;
Tfin=Te*Nb_vec
sigma=0.2;
```

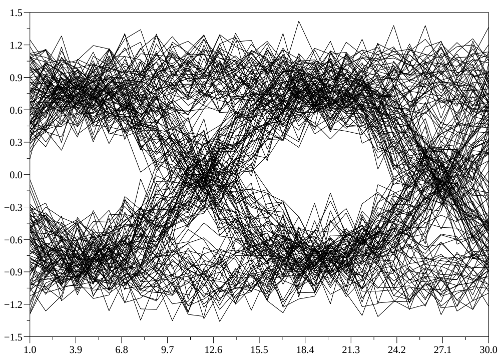
3.1.3 Scope Results



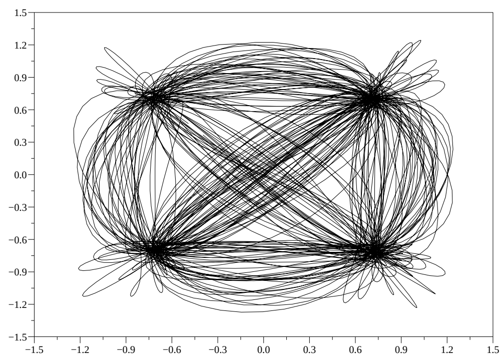
(a)



(b)



(c)



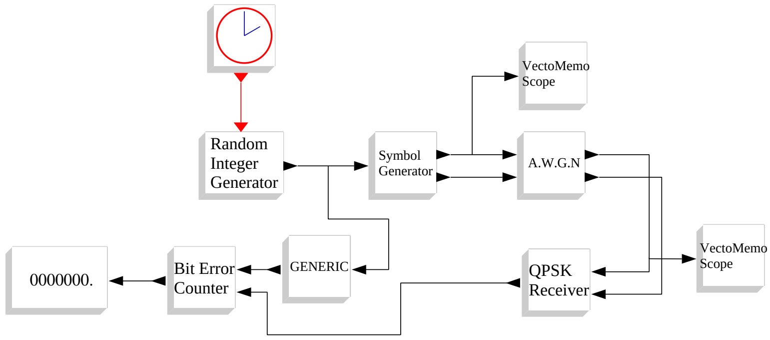
(d)

3.1.4 Mod_num blocks

- VECTOMEMOSCOPE_f **-** Vectorial scope with memory block
- SCOXYVEC_f **-** Scicos Vector Scope visualization block
- CONVOLGEN_f **-** Convolution block
- INTSYMB_f **-** Discrete Symbol Integrator block

- MODPSK_f - M-ary Phase Shift Keying Modulator Block
- DEMODPSK_f - M-ary Phase Shift Keying demodulator Block
- OVERLAPRSR_f - Memory discrete shift register for multiplexed signal block
- NOISEBLK_f - Gaussian White Noise Generator block
- GENINT_f - Random Integer Generator block
- BITERROR_f - Binary error estimation block
- COMP_f - One bit Quantizer block
- UPSMPL_f - Up-sample block
- DOWNSMPL_f - Down-sample block

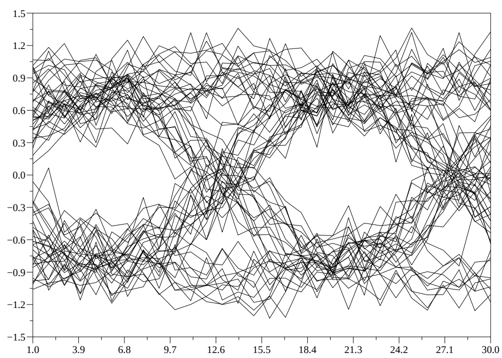
3.2 Integrated diagram of base band QPSK transmission chain



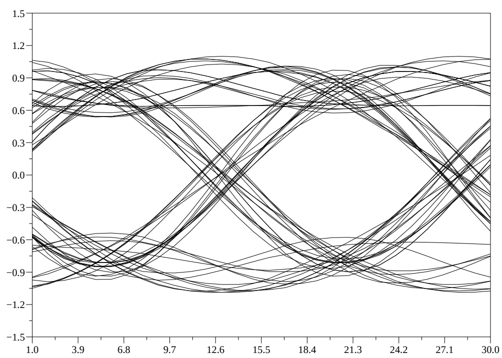
3.2.1 Context

```
Te=0.1;
Nu=128;
Nech=15;
N=Nu*Nech;
Nbit=2;
nb_coef=127;
r=0.35;
fe=Nech;
gain_em=1;
gain_rec=1/Nech;
p=filter_tap(1,nb_coef,Nech,r,1);
m1=2^(int(log(N+nb_coef-1)/log(2))+1);
Nb_vec=3;
nb_event=3
Tfin=Te*Nb_vec;
sigma=0.2;
```

3.2.2 Scope Results



Scope results



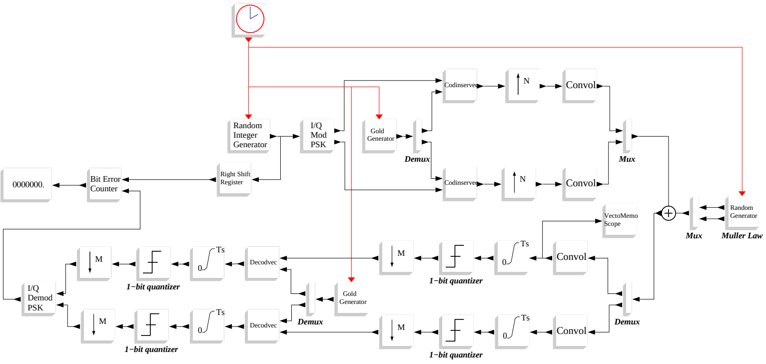
Scope results

3.2.3 Mod_num blocks

- AWGN_f - Additive White Gaussian Noise Channel block
- GENINT_f - Random Integer Generator block
- GENSYMB_f - Symbol generator block
- QPSKREC_f - Quaternary Phase Shift Keying receiver block

- BITERROR_f - Binary error estimation block
- VECTOMEMOSCOPE_f - Vectorial scope with memory block

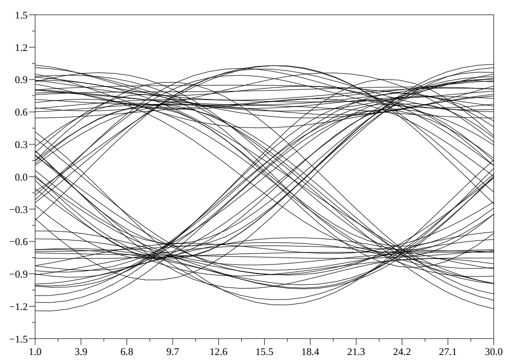
3.3 Base band single user QPSK spread-spectrum transmission chain



3.3.1 Context

```
Te=0.1;
Nu=4;
Nbit=2;
l_g=5;
Nc=(2^l_g)-1;
ci_g1=[21;21];ci_g2=[13;10]
coef_g1=[9;9];coef_g2=[15;15]
nb_coef=127;
Nech=15;
N=Nu*Nc;
N1=N*Nech
r=0.35;
fe=Nech;
gain_em=1;
gain_rec=1/Nech;
pulse=zeros(1,1);
t_Ts=1/fe*(-nb_coef/2:nb_coef/2-1);
pulse = 4*r/%pi*(cos((1+r)*%pi*t_Ts) + (sin((1-r)*%pi*t_Ts)./(4*r*t_Ts)))./(1-(4*r*t_Ts).^2);
Nb_vec=3;
nb_event=4;
Tfin=Te*Nb_vec;
sigma=0.4;
```

3.3.2 Scope Results



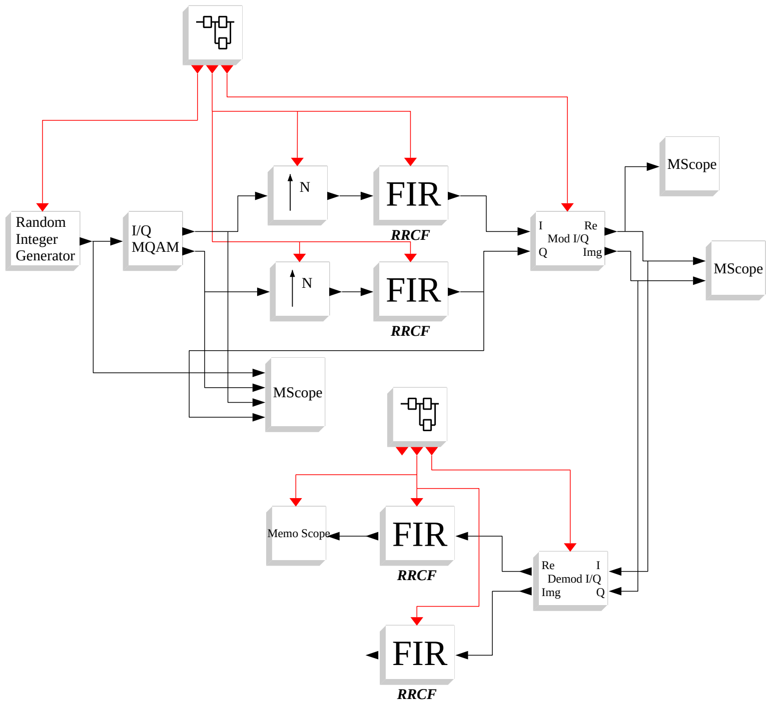
Scope results

3.3.3 Mod_num blocks

- CODINSERVEC_f - Sequence/symbol modulation block

- DECODVEC_f - Sequence/symbol demodulation block
- GENGOLD_f - Gold sequence generator block
- CONVOLGEN_f - Convolution block
- DEMODPSK_f - M-ary Phase Shift Keying demodulator Block
- OVERLAPRSR_f - Memory discrete shift register for multiplexed signal block
- MODPSK_f - M-ary Phase Shift Keying Modulator Block
- INTSYMB_f - Discrete Symbol Integrator block
- GENINT_f - Random Integer Generator block
- NOISEBLK_f - Gaussian White Noise Generator block
- VECTOMEMOSCOPE_f - Vectorial scope with memory block
- BITERROR_f - Binary error estimation block
- COMP_f - One bit Quantizer block
- UPSMPL_f - Up-sample block
- DOWNSMPL_f - Down-sample block

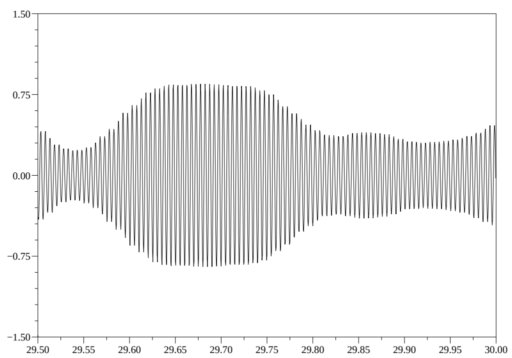
3.4 Scattered diagram of sample-based 16-QAM chain transmission



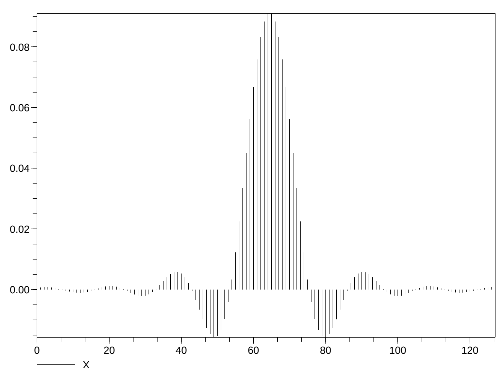
3.4.1 Context

```
Tsymbole = 0.1
Nech = 12
Tech = Tsymbole /Nech
Nsampl = 20
Tsampl = Tech/Nsampl

nb_coef=127
r=0.35

wo=2*%pi/(Tsymbole/20)
Tfin=300*Tsymbole
```

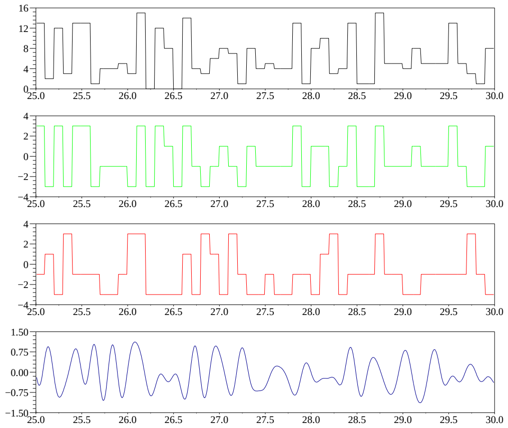
3.4.2 Scope Results



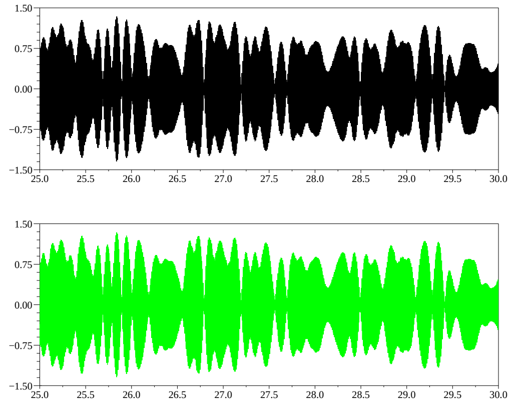
Scope results



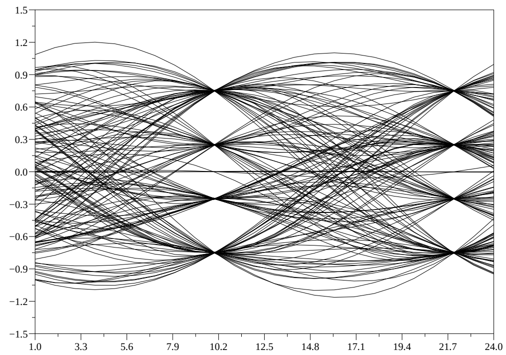
Scope results



Scope results



Scope results

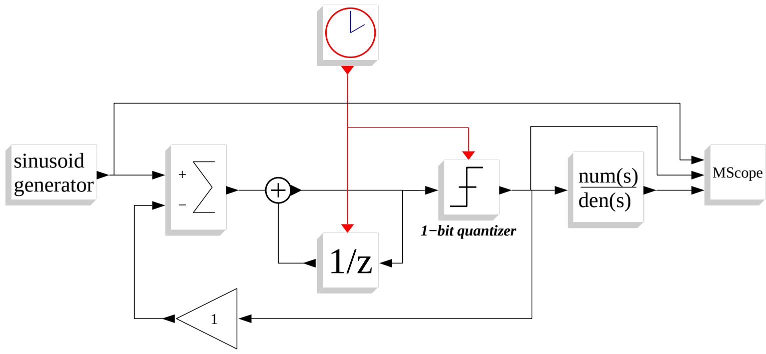


Scope results

3.4.3 Mod_num blocks

- GENINT_f - Random Integer Generator block
- RIFGEN_f - Generic Finite Impulse Response filter block
- MODQAM_f - Quadrature Amplitude Modulation modulator Block
- MODIQ_f - I/Q modulator Block
- DEMODIQ_f - I/Q demodulator Block
- MEMOSCOPE_f - Memory sequential scope block
- UPSMPL_f - Up-sample block

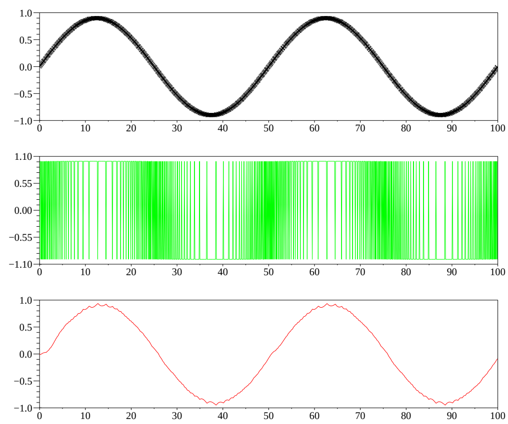
3.5 Scattered diagram of first order Delta-Sigma modulator



3.5.1 Context

```
M=1
Te=0.1
Ne=500
To=Te*Ne
w=2*%pi/To
Tfin=2*To
```

3.5.2 Scope Results



Scope results

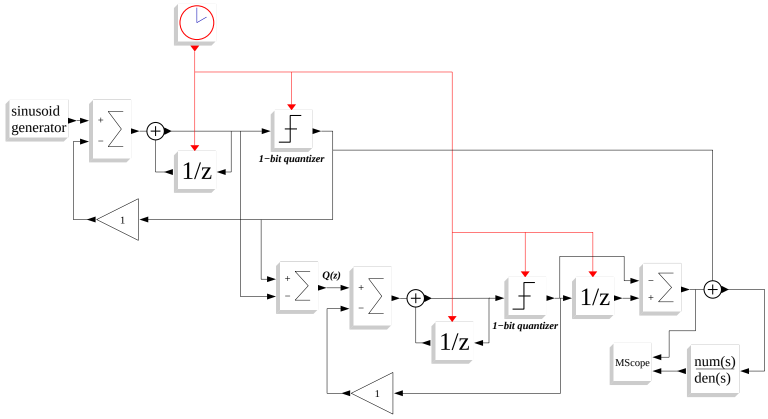
3.5.3 Mod_num blocks

- COMP_f - One bit Quantizer block

3.5.4 See Also

- mash_2eme_ordre - Scattered diagramm of second order Delta-Sigma modulator (Scicos Diagram)
- mash_gauss - Integrated diagram of Delta-Sigma modulator (Scicos Diagram)
- MASHBLK_f - Delta-Sigma modulator block (Scicos Block)
- mash_gauss_sim - Output spectra of Delta-Sigma modulator (Scilab simulation Script)

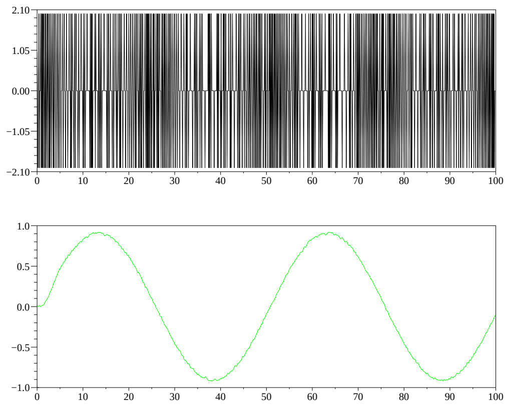
3.6 Scattered diagramm of second order Delta-Sigma modulator



3.6.1 Context

```
M=1
Te=0.1
Ne=500
To=Te*Ne
w=2*%pi/To
Tfin=2*To
```

3.6.2 Scope Results



Scope results

3.6.3 Mod_num blocks

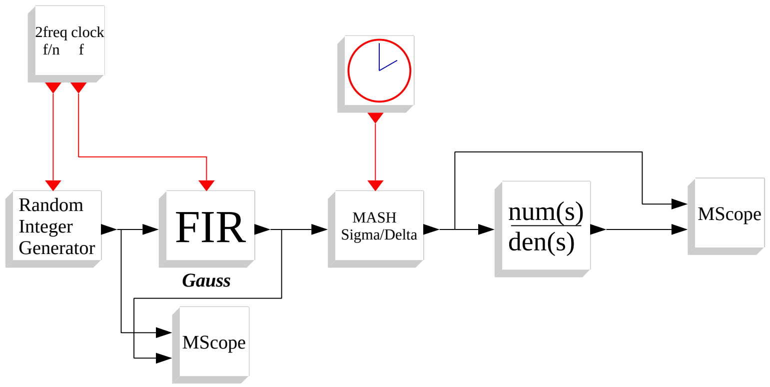
- COMP_f - One bit Quantizer block

3.6.4 See Also

- mash_1er_ordre - Scattered diagram of first order Delta-Sigma modulator (Scicos Diagram)
- mash_gauss - Integrated diagram of Delta-Sigma modulator (Scicos Diagram)
- MASHBLK_f - Delta-Sigma modulator block (Scicos Block)

- mash_gauss_sim - Output spectra of Delta-Sigma modulator (Scilab simulation Script)

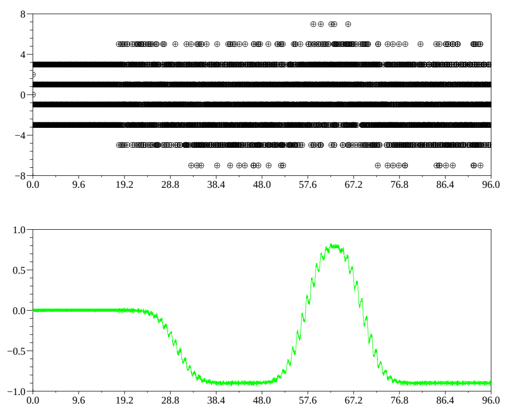
3.7 Integrated diagram of Delta-Sigma modulator



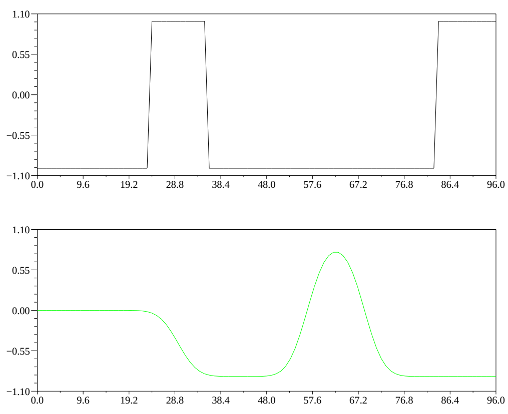
3.7.1 Context

```
M=1
Te=1
Nech=12

order=3
Tsym=Te*Nech

Nsampl=500
Tsampl=Tsym/Nsampl

fc=0.03*(1/Tsampl)
eps=0.7
tau=1/(2*%pi*fc)

Tfin=8*Tsym
```

3.7.2 Scope Results



Scope results



Scope results

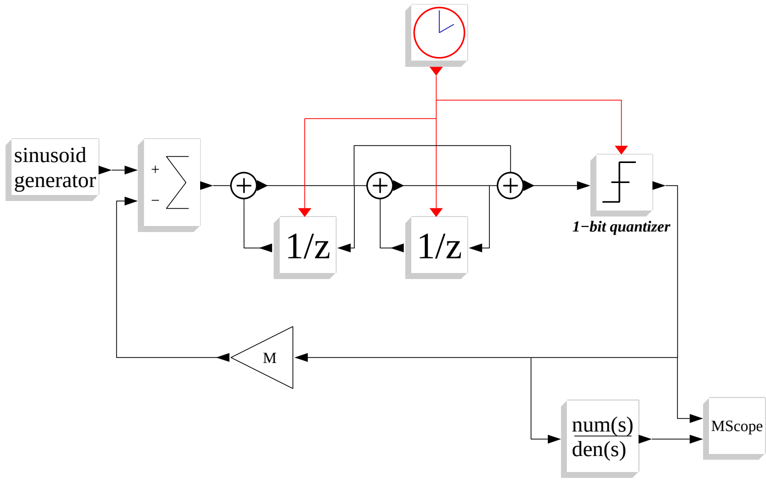
3.7.3 Mod_num blocks

- RIFGEN_f - Generic Finite Impulse Response filter block
- GENINT_f - Random Integer Generator block
- MASHBLK_f - Delta-Sigma modulator block

3.7.4 See Also

- mash_1er_ordre - Scattered diagram of first order Delta-Sigma modulator (Scicos Diagram)
- mash_2eme_ordre - Scattered diagramm of second order Delta-Sigma modulator (Scicos Diagram)
- mash_gauss_sim - Output spectra of Delta-Sigma modulator (Scilab simulation Script)

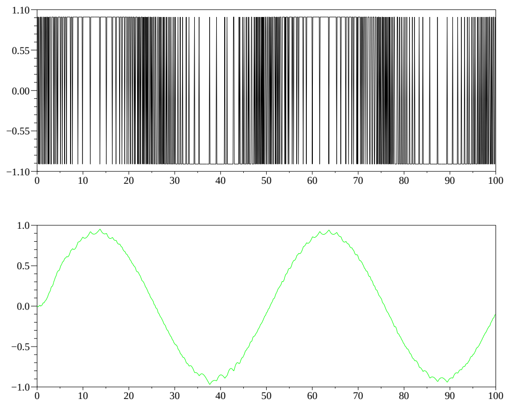
3.8 Delta-sigma modulator with two in-loop integrators



3.8.1 Context

```
M=1
Te=0.1
Ne=500
To=Te*Ne
w=2*%pi/To
Tfin=2*To
```

3.8.2 Scope Results

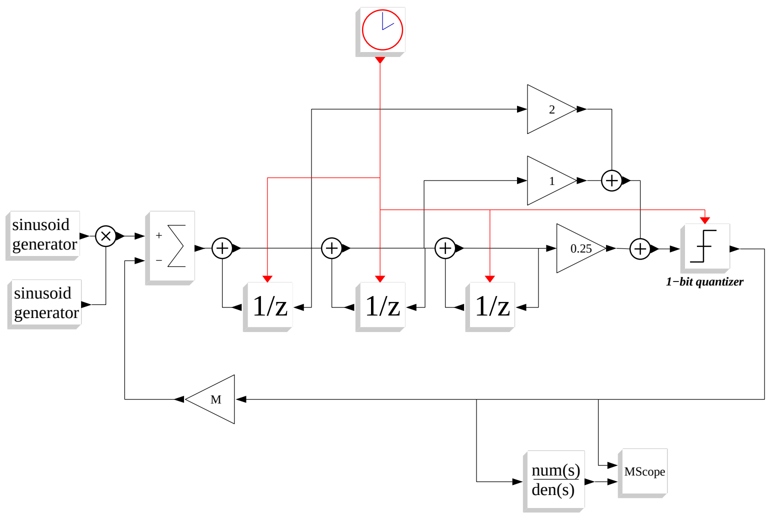


Scope results

3.8.3 Mod_num blocks

- COMP_f - One bit Quantizer block

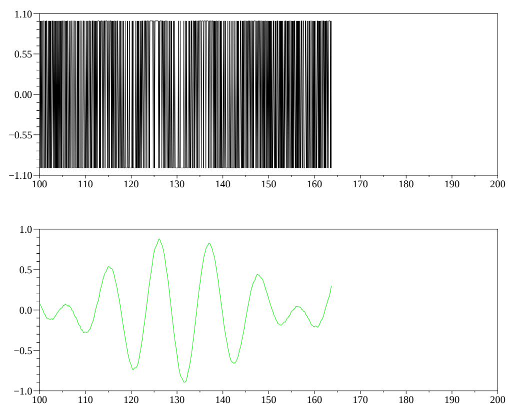
3.9 Delta-sigma modulator with three in-loop integrators



3.9.1 Context

```
M=1
Te=1
Ne=100
To=Te*Ne
w=2*%pi/To
Tsampl=Te/Ne
Nsampl=2^14
Tfin=Nsampl*Tsampl
```

3.9.2 Scope Results

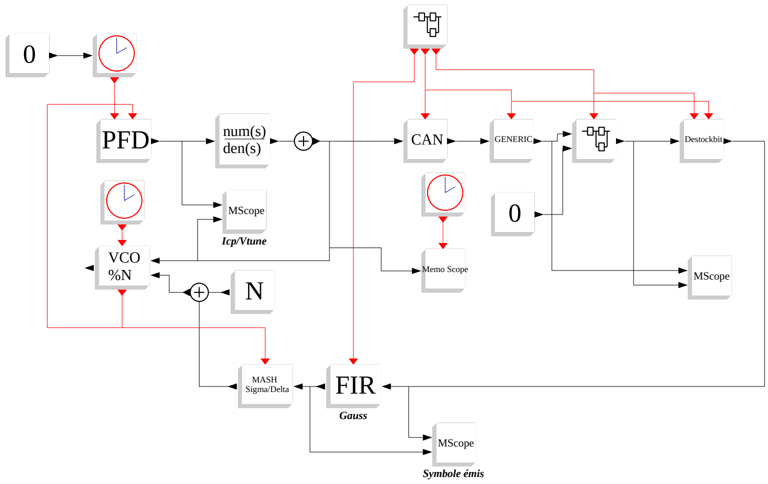


Scope results

3.9.3 Mod_num blocks

- COMP_f - One bit Quantizer block

3.10 Chaotic frequency hopping emitter



3.10.1 Description

Add here a paragraph of the function description.

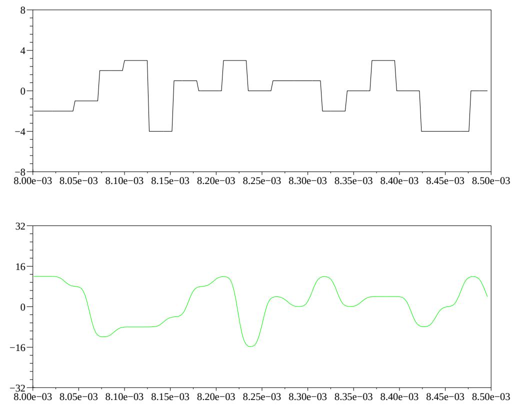
3.10.2 Context

```
nb_coef=85
lines(-1);
F_cpf = 20e6;
T_cpf = 1/F_cpf;
sig_ref=T_cpf*0.02/100

Fo = 2.045e9;
To = 1 / Fo;
wo=2*%pi * Fo;
kv = 100.5e6;
alpha=6.91e9;
beta2=0.15;
j_vco=0.2e6;

Fd=2.5e9;
N=int(Fd/F_cpf)

Nsampl = 2;
Tsampl = 1/(F_cpf*Nsampl);
Fsampl = 1 / Tsampl;

Icp = 5e-3;
Ileak=0.1e-6;
fn=F_cpf/180;
phi=%pi/4;

Ts=3/fn
Nech=12
Te=Ts/Nech

M=32
Nbit=3
a=0.5
b=-1

[tau1,tau,tau2]=calcul_3eme_ordre(fn,phi,kv*2*%pi,Icp,N);
s=poly(0,'s');
num=1+tau1*s;
den=tau*s*(1+tau2*s);
Tfin=170e3*T_cpf
```

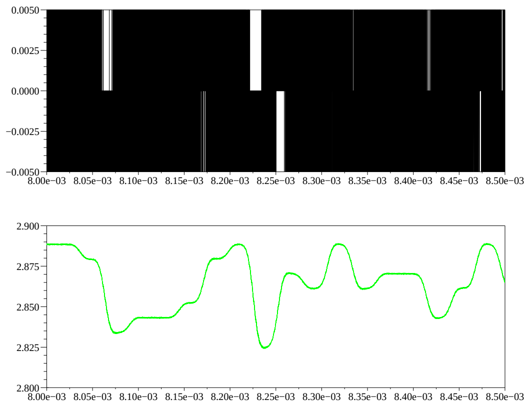
3.10.3 Scope Results

3.10.4 Mod_num blocks

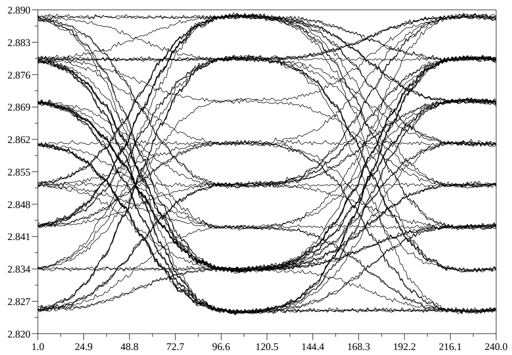
- PFD_f - Phase/Frequency tristate Detector block
- VCO_f - Discrete Voltage Controlled Oscillator block
- PCLOCK_f - Integrated phase modulator event generator block



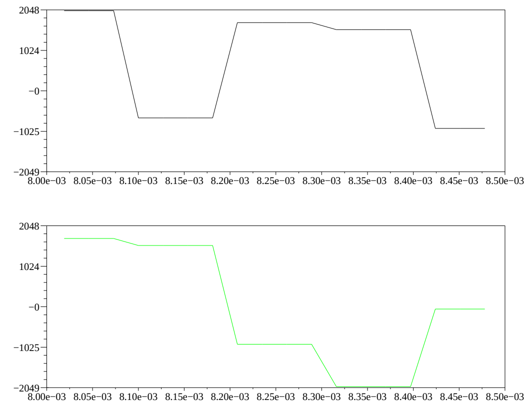
Scope results



Scope results



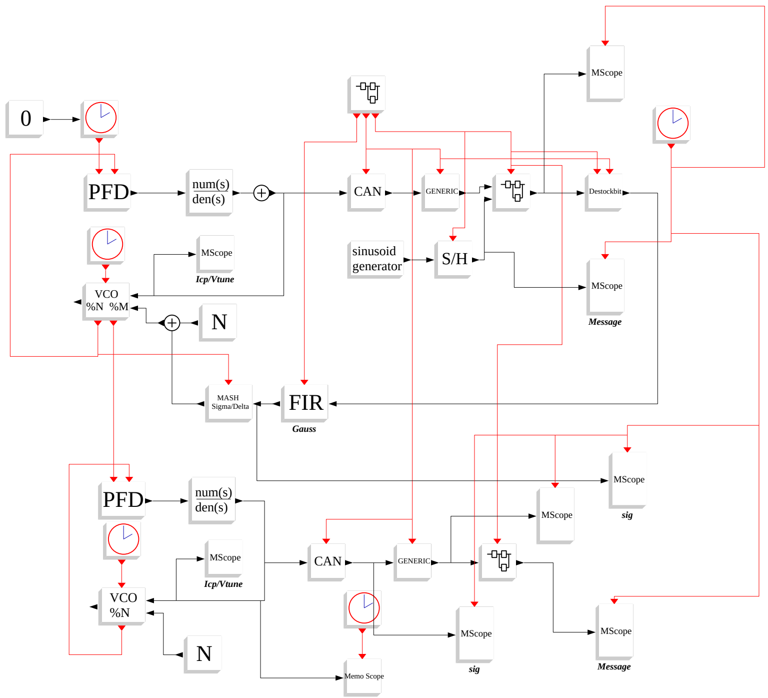
Scope results



Scope results

- MASHBLK_f - Delta-Sigma modulator block
- RIFGEN_f - Generic Finite Impulse Response filter block
- CAN_f - Analog to Digital Converter block
- MEMOSCOPE_f - Memory sequential scope block
- DESTOCKBIT_f - Binary shift register block

3.11 Chaotic frequency hopping transmission system



3.11.1 Description

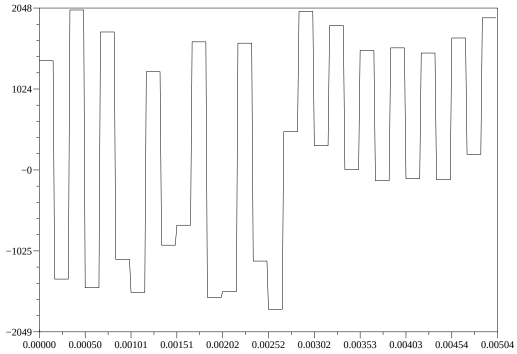
Add here a paragraph of the function description.

3.11.2 Context

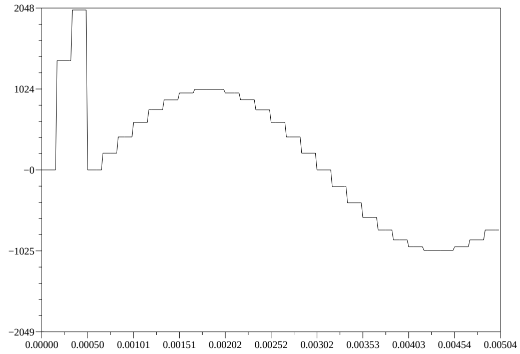
```
nb_coef=85
lines(-1);
F_cpf = 20e6;
T_cpf = 1/F_cpf;
sig_ref=T_cpf*0.02/100

Fo = 2.045e9;
To = 1 / Fo;
wo=2*%pi * Fo;
kv = 100.5e6;
alpha=6.91e9;
beta2=0.15;
j_vco=0.2e6;

Fd=2.5e9;
N=int(Fd/F_cpf)

Nsampl = 2;
Tsampl = 1/(F_cpf*Nsampl);
Fsampl = 1 / Tsampl;

Icp = 5e-3;
Ileak=0.1e-6;
fn=F_cpf/280;
phi=%pi/4;

Ts=3/fn
Nech=12
Te=Ts/Nech

M=32
Nbit=3
a=0.5
b=-1

[tau1,tau,tau2]=calcul_3eme_ordre(fn,phi,kv*2*%pi,Icp,N);
s=poly(0,'s');
num=1+tau1*s;
den=tau*s*(1+tau2*s);
fn2=F_cpf/100;
[tau1,tau,tau2]=calcul_3eme_ordre(fn2,phi,kv*2*%pi,Icp,N);
```

```
num2=1+tau1*s;
den2=tau*s*(1+tau2*s);
Tfin=0.00504
```

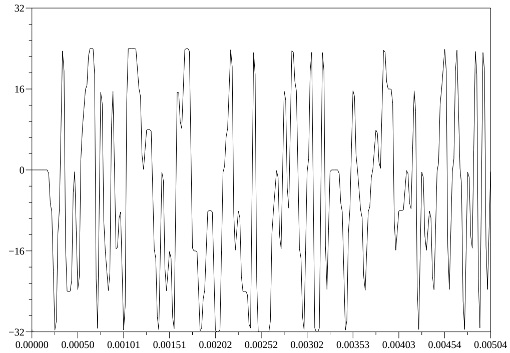
3.11.3 Scope Results



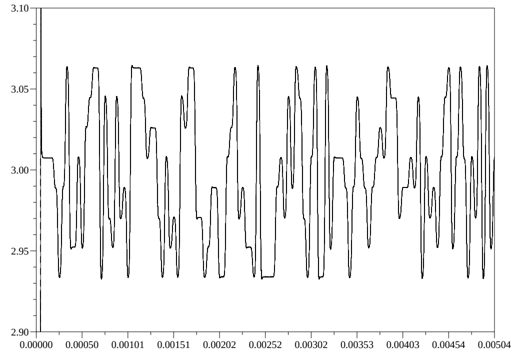
Scope results



Scope results



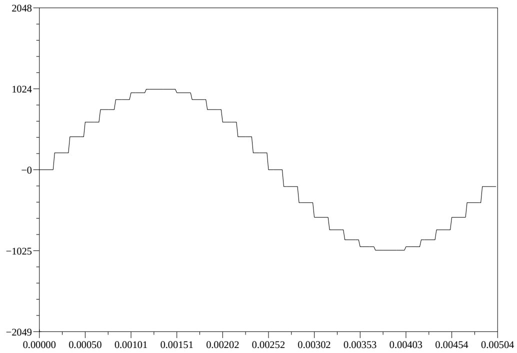
Scope results



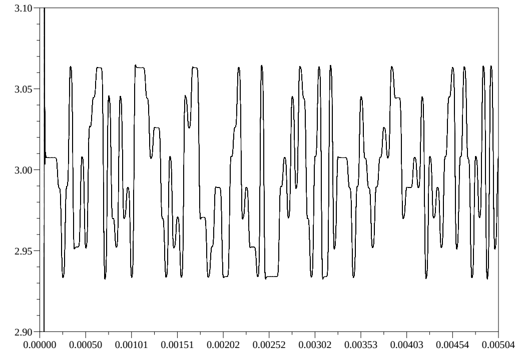
Scope results

3.11.4 Mod_num blocks

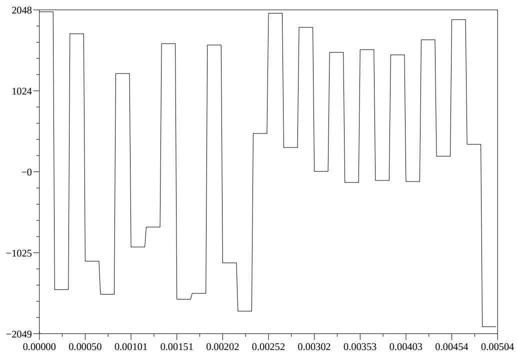
- PFD_f - Phase/Frequency tristate Detector block
- VCO_f - Discrete Voltage Controlled Oscillator block
- PCLOCK_f - Integrated phase modulator event generator block
- MASHBLK_f - Delta-Sigma modulator block
- RIFGEN_f - Generic Finite Impulse Response filter block
- CAN_f - Analog to Digital Converter block
- DESTOCKBIT_f - Binary shift register block



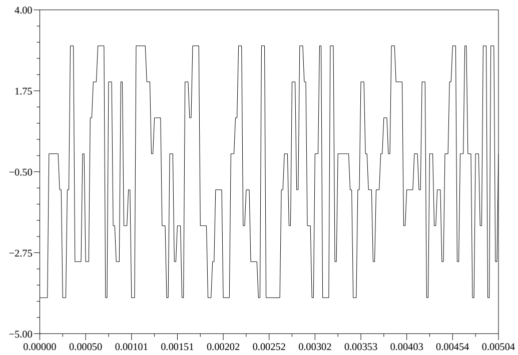
Scope results



Scope results



Scope results



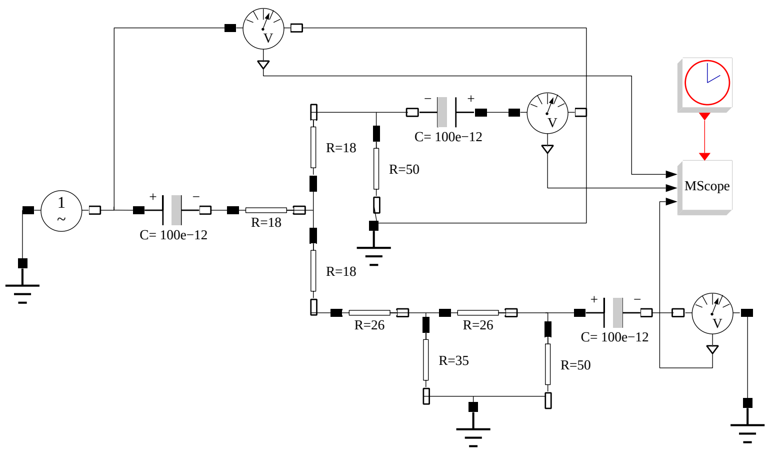
Scope results

- MEMOSCOPE_f - Memory sequential scope block

Chapter 4

Electrical circuits

4.1 Resitif attenuator/coupler



4.1.1 Description

This diagram is a power attenuator realised with resistors.

This stage is often find in microwave synthesizer at the power output of the VCO.

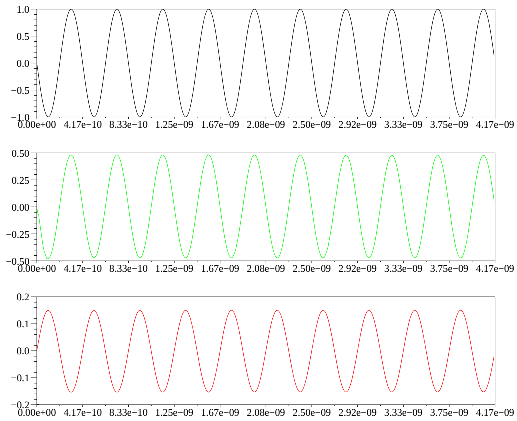
4.1.2 Context

```
do_tild=mdo_tild
%scicos_solver=100
f_sin=2.4e9;
T_sin=1/f_sin;
Nsampl=50;
Tsampl=T_sin/Nsampl; f_sin=2.4e9;
Tfin=10*T_sin
```

4.1.3 Scope Results

4.1.4 Mod_num blocks

- mResistor - Modified Resistor block



Scope results

Part II

Simulation Scilab scripts

Chapter 1

Simulations of chaotic systems

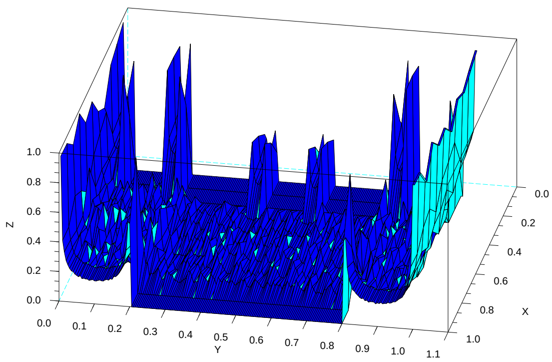
1.1 Output probability density of autonomous second order IIR filter

1.1.1 Simulation script(s)

```
function [x,y]=mhistplot(n,data)
  p=size(data,'*')
  data=data(:)

  if size(n,'*')==1 then
     x = linspace(min(data), max(data), n+1)';
  else
     x=n(:)
  end,
  n=prod(size(x));

  [ind , y] = dsearch(data, x);

  y=[y;y(n-1)];
  nx=maxi(min(15,prod(size(x))-1),1);
endfunction

set("figure_style","old")
load SCI/macros/scicos/lib
exec(loadpallibs,-1)
%tcur=0;%cpr=list();alreadyran=%f;needstart=%t;needcompile=4;%state0=list();
prot=funcprot();funcprot(0);
deff('disablemenus()',' ')
deff('enablemenus()',' ')
funcprot(prot)

load(MODNUM+'/simu/lin_chua_sim/lin_chua_sim.cos')

context=scs_m.props("context");
execstr(context);

tolerances=scs_m.props.tol;
solver=tolerances(6);
scs_m.props.context(3)='Nbit=16';
ci=0.66:2e-3:0.68;
Nbsampl=1e4;
Nbsampl=5e3;
z=zeros(size(ci,2),Nbsampl);
for j=1:size(ci,2)
 scs_m.props.context(6)='ci='+string(ci(j));
```

```
 %scicos_context=struct();
 [%scicos_context,ierr]=script2var(scs_m.props.context,%scicos_context);
 [scs_m,%cpr,needcompile,ok]=do_eval(scs_m,%cpr);
 [%cpr,%state0_n,needcompile,alreadyran,ok]=do_update(%cpr,%state0,needcompile);
 %state0=%state0_n;
 %tcur=0;
 %cpr.state=%state0;
 tf=scs_m.props.tf;
 [state,t]=scicosim(%cpr.state,%tcur,tf,%cpr.sim,'start',tolerances);

 timer();
 y=[];
 %tcur=0;
 for i=1:Nbsampl

   tf=i*Te;
   [state,t]=scicosim(%cpr.state,%tcur,tf,%cpr.sim,'run',tolerances);
   %cpr.state=state;
   %tcur=tf;

   Nblock=3;
   y(1,i)=state("z")(%cpr.sim("zptr")(Nblock):%cpr.sim("zptr")(Nblock+1)-1);
   Nblock=6;
   y(2,i)=state("z")(%cpr.sim("zptr")(Nblock):%cpr.sim("zptr")(Nblock+1)-1);

 end
 z(j,:)=y(1,:);
 Tsim=timer();disp('Tsim:'+string(Tsim));

end

clear y;
k=200;
c=zeros(size(ci,2),k+1);
for j=1:size(ci,2)
 [a,b]=mhistplot(k,z(j,:));
 c(j,:)=matrix(b,1,k+1);
end
x=(1:size(ci,2))/size(ci,2);
y=(1:k+1)/k;
z=c/max(c);
set("figure_style","new");
scf(100);
plot3d(x,y,z,11,45,'X@Y@Z',[2,2,4])
```

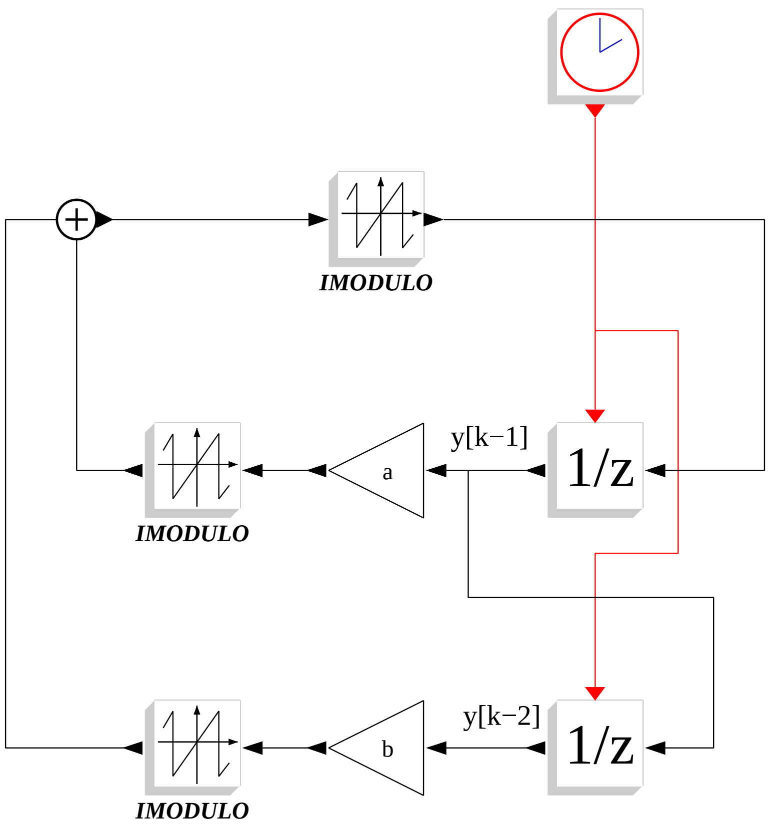
lin_chua_sim.sce



Scope results

1.1.2 Scope Results

1.1.3 Scicos diagram(s)



lin_chua_sim.cos

1.1.4 Mod_num blocks

- IMODULO_f - Integer modulo function block

1.1.5 See Also

- lin_chua - Second order Infinite Impulse Response filter (Scicos Diagram)

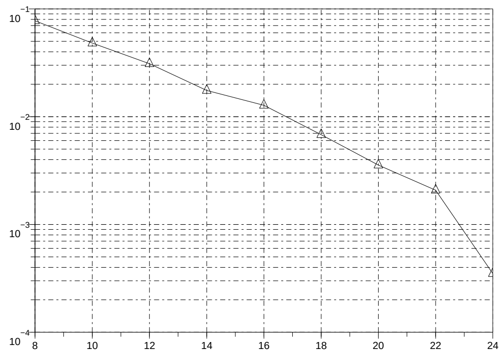
1.2 BER estimation of pair chaotic encoder/decoder with introduction of jitter in receiver clock

1.2.1 Simulation script(s)

```
//Define simulation name
sim_name='lin_chua_teb_sim';

//Load scicos diagram
load(MODNUM+'/simu/'+sim_name+'/'+sim_name+'.cos')

//Define context
exec(MODNUM+'/simu/'+sim_name+'/'+sim_name+'_ctxt.sce');
context=scs_m.props("context");execstr(context);

//Define Simulation end time
scs_m.props.tf=Tfin;

//Define other variable
//sig_log=10:2.5:32.5;
sig_log=8:2:24;
sig=[];
for k=1:size(sig_log,2)
  sig(k)=1/(10^(sig_log(k)/10));
end
nb_r=size(sig,1);
teb=[];nb_error=[];
l=0

while l<2
  for j=1:nb_r
    //define context variable to be sweep
    scs_m.props.context(13)='sigma='+string(sig(j))+'*Te';
    //Initialise Info and %scicos_context variable
    Info=list();
    if ~exists('%scicos_context') then %scicos_context=struct(); end

    //Disp Begin simulation date
    printf("Begin simulation %d, \t %s\n",j,return_date());

    //Do simulation with scicos_simulate
    Info=scicos_simulate(scs_m,Info,%scicos_context,"nw");

    //Save result
    myvar=return_state_block(Info,"toto");
    nb_error(j)=myvar(1)(1);

    //Processed result
    teb(j)=nb_error(j)/(Nbsampl*Nbit);

    //Disp processed result and end simulation time
    printf("Processed Teb(%d) = %f\n",j,teb(j));
    printf("End simulation %d, \t %s\n\n",j,return_date());
  end
  l=l+1
end

//Plot result
plot_teb(49,sig_log,teb,"1");
```

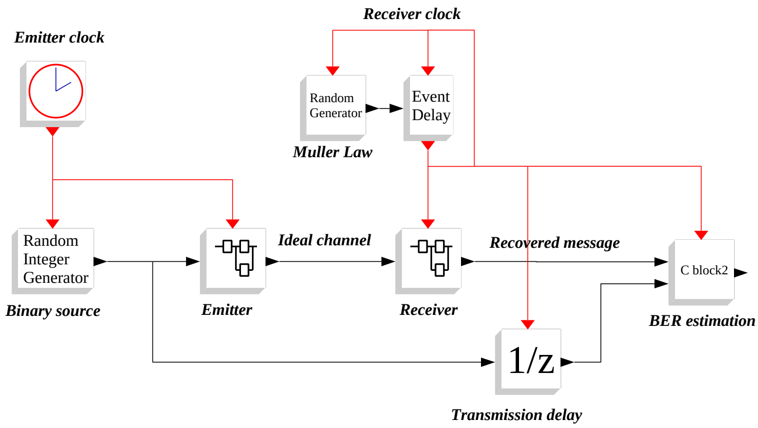
lin_chua_teb_sim.sce

1.2.2 Scope Results



Scope results

1.2.3 Scicos diagram(s)



lin_chua_teb_sim.cos

1.2.4 Context file(s)

```
scs_m.props.context=[
 'Te=1;'
 'Nbsampl=2^16;'
 'Tfin=Nbsampl*Te;'
 'Nbit=16;'
 'a=0.5;'
 'b=-2.5;'
 'ci=0.675;'
 'Intmax = 2^(Nbit-1);'
 'ci1=-int(ci*Intmax);'
 'ci2=int(ci*Intmax);'
 'p_source=2*%pi/(4*Te);'
```

```
'a_source=2^(Nbit-2);'
'sigma=0'];
```

lin_chua_teb_sim_ctxt.sce

1.2.5 Mod_num blocks

- GENINT_f - Random Integer Generator block
- NOISEBLK_f - Gaussian White Noise Generator block

Chapter 2

Simulations of oscillators and Phase Locked Loops

2.1 Output spectra of Integer N frequency synthesizer

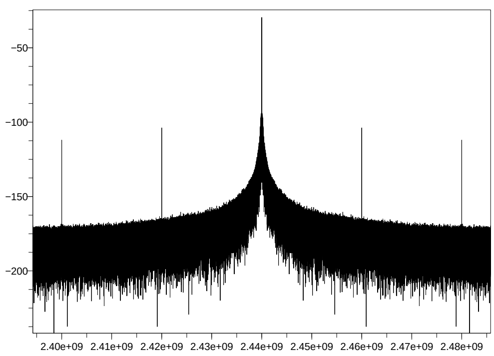
2.1.1 Simulation script(s)

```
//Define simulation name
sim_name='synthe_int_sim';
//Define simulation path
sim_path='synthe_int_sim';
//Load scicos diagram
load(MODNUM+'/simu/'+sim_path+'/'+sim_name+'.cos');
//Define context
exec(MODNUM+'/simu/'+sim_path+'/'+sim_name+'_ctxt.sce');
context=scs_m.props("context");execstr(context);
//Define Simulation end time
scs_m.props.tf=Tfin;
//Define other variables

//Open Log file
u_log=mopen(MODNUM+'/simu/'+sim_path+'/'+sim_name+'.log','a');
//Set flag (for display saved information in scilab window)
flag=1;
//Save inital state of simulation in Log file
wlog_init(u_log,flag);
//Save context in log file
wlog_ctxt(u_log,flag);
//Loop for iterative simulation
j=1;
//Save and display begin simulation date j
wlog_bst(u_log,j,flag);
//substitue context variable to be sweep
//scs_m.props.context=subst_ctxt('varname','varname=');
//Initialise Info and %scicos_context variable
//if exists('Info')==0 then
clear Info;
Info=list();

//end
//if exists('%scicos_context')==0 then %scicos_context=struct(); end
clear %scicos_context;
%scicos_context=struct();
//Do simulation with scicos_simulate
Info=scicos_simulate(scs_m,Info,%scicos_context);
```

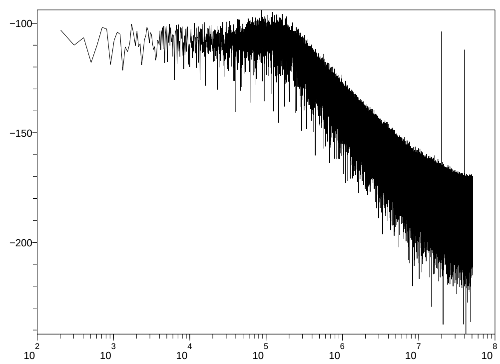
```
//Load result
myvar=return_state_block(Info,"z_buf");
//Save and display util variable
//wlog_sv(u_log,tlist(['varname'],varvalue),flag);
//Save and Disp final simulation date j
wlog_fst(u_log,j,flag);
//post-processed result
//Save and display post-processed vector(matrix)
//wlog_psv(u_log,tlist(['varname'],varvalue,flag);
//Close final state of simulation in Log file
wlog_final(u_log,flag);
//close log file
mclose(u_log);

plot_spectre2(myvar(1)(1:Nsav),Tsampl,Tacqui,N*F_cpf,0.9e6,10,0);
plot_spectre2(myvar(1)(1:Nsav),Tsampl,Tacqui,N*F_cpf,0.5e6,11,1);
```

synthe_int_sim.sce

2.1.2 Scope Results

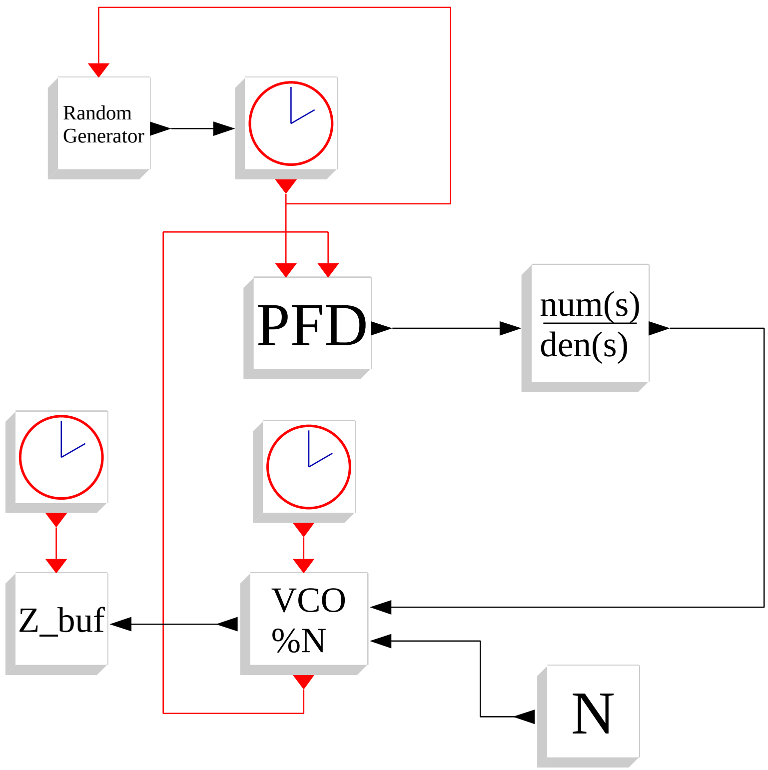


(a) Output spectrum of synthesizer



(b) Output noise spectrum of synthesizer

2.1.3 Scicos diagram(s)



synthe_int_sim.cos

2.1.4 Context file(s)

```
scs_m.props.context=[
'F_cpf = 20e6;'
'T_cpf = 1/F_cpf;'
'sig_ref=(2*%pi*0.01/100)'
'Fo = 2.045e9;'
'To = 1 / Fo;'
'wo=2*%pi * Fo;'
'kv = 100.5e6;'
'alpha=6.91e9;'
'beta2=0.15;'
'Fd=2.45e9;'
'N=int(Fd/F_cpf)'
'j_vco=2e6/N;'
'Nsampl = 16;'
'Tsampl = T_cpf/Nsampl;'
'Fsampl = 1 / Tsampl;'
'Icp = 5e-3;'
'Ileak=10e-6;'
'Ileak_sig=1e-12;'
'fn=F_cpf/180;'
'phi=%pi/4;'
'[tau1,tau,tau2]=calcul_3eme_ordre(fn,phi,kv*2*%pi,Icp,N);'
's=poly(0,''s'');'
'num=1+tau1*s;'
```

```
'den=tau*s*(1+tau2*s);'
'Nsav=2^21+2^20'
'Tse=Tsampl'
'Tacqui=1e-4'
'Tfin=Tacqui+Nsav*Tse'
];
```

synthe_int_sim_ctxt.sce

2.1.5 Mod_num blocks

- PFD_f - Phase/Frequency tristate Detector block
- VCO_f - Discrete Voltage Controlled Oscillator block
- PCLOCK_f - Integrated phase modulator event generator block
- NOISEBLK_f - Gaussian White Noise Generator block
- ZBUF_f - Discrete buffer block

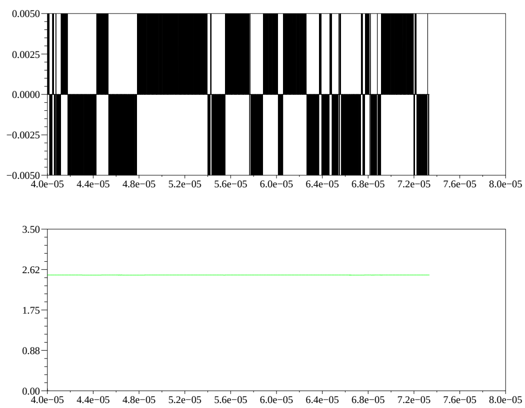
2.2 Output jitter of integer-N frequency synthesizer

2.2.1 Simulation script(s)

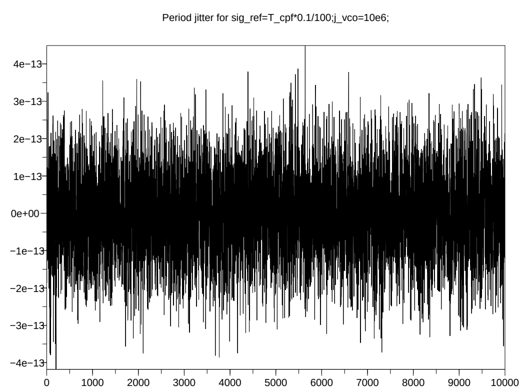
```
//Define simulation name
sim_name='synthe_int_jit_sim';
//Define simulation path
sim_path='synthe_int_jit_sim';
//Load scicos diagram
load(MODNUM+'/simu/'+sim_path+'/'+sim_name+'.cos');
//Define context
exec(MODNUM+'/simu/'+sim_path+'/'+sim_name+'_ctxt.sce');
context=scs_m.props("context");execstr(context);
//Define Simulation end time
scs_m.props.tf=Tfin;
//Define other variables

j=1;
//substitue context variable to be sweep
//scs_m.props.context=subst_ctxt('varname','varname=');
//Initialise Info and %scicos_context variable
clear Info;Info=list();
clear %scicos_context;
%scicos_context=struct();
//Do simulation with scicos_simulate
Info=scicos_simulate(scs_m,Info,%scicos_context);
//Load result
myvar=return_state_block(Info,"z_buft");

f=scf(10);
plot2d(1:Nsav,myvar(1)(1:Nsav)-1/Fd,rect=[0 min(myvar(1)(1:Nsav)-1/Fd) Nsav max(myvar(1)(1:N
xtitle('Period jitter for '+scs_m.props.context(3)+';'+scs_m.props.context(12));
g=scf(20);
histplot(100,myvar(1)(1:Nsav)-1/Fd);
xtitle('Probility density of period jitter for '+scs_m.props.context(3)+';'+scs_m.props.cont
```

synthe_int_jit_sim.sce

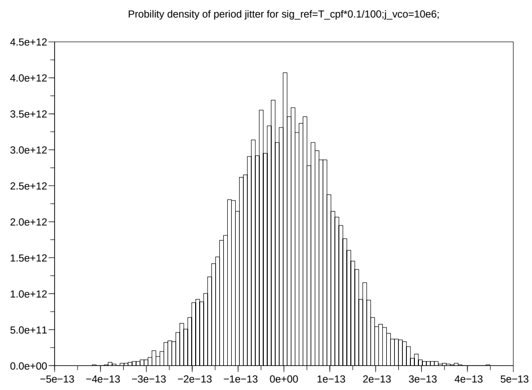
2.2.2 Scope Results



(a) Charge pump output current; input voltage of VCO

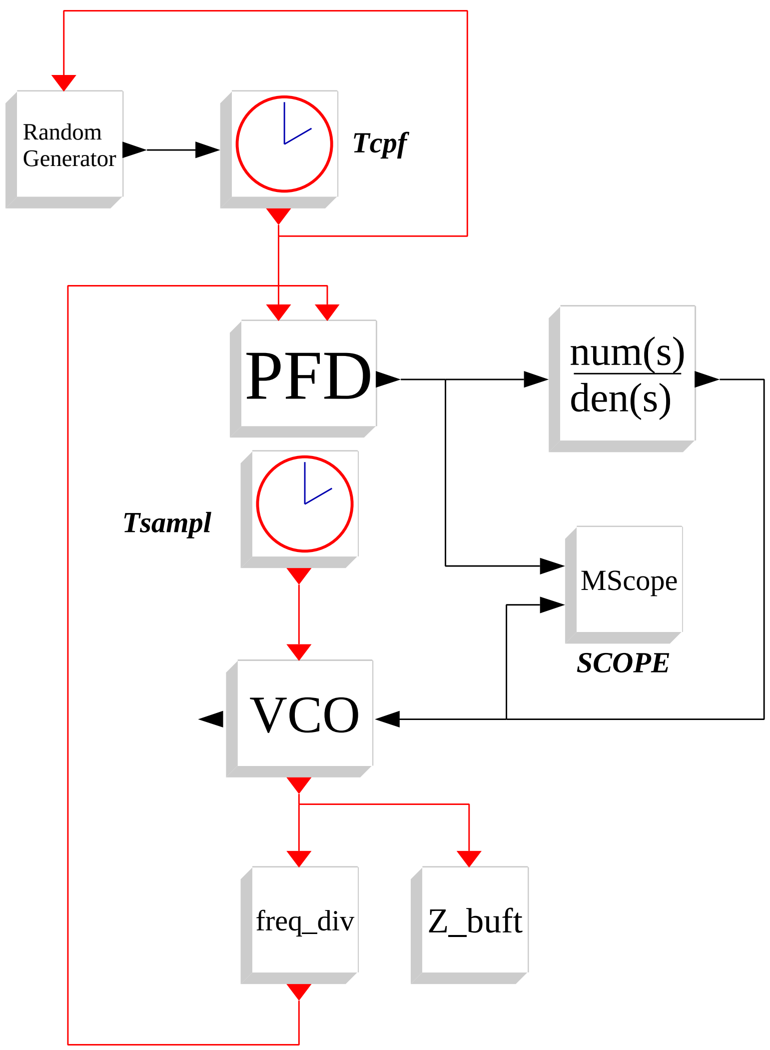


(b) Evolution of period jitter



(c) Probability density of period jitter

2.2.3 Scicos diagram(s)



synthe_int_jit_sim.cos

2.2.4 Context file(s)

```
scs_m.props.context=[
'F_cpf = 20e6;'
'T_cpf = 1/F_cpf;'
'sig_ref=T_cpf*0.1/100'
'Fo = 2.045e9;'
'To = 1 / Fo;'
'wo=2*%pi * Fo;'
'kv = 100.5e6;'
'alpha=6.91e9;'
'beta2=0.15;'
'Fd=2.44e9;'
'N=int(Fd/F_cpf)'
'j_vco=10e6;'
'Nsampl = 2;'
'Tsampl = To/Nsampl;'
'Fsampl = 1 / Tsampl;'
'Icp = 5e-3;'
'Ileak=0;'
'Ileak_sig=1e-6;'
'fn=F_cpf/180;'
'phi=%pi/4;'
'[tau1,tau,tau2]=calcul_3eme_ordre(fn,phi,kv*2*%pi,Icp,N);'
's=poly(0,''s'');'
'num=1+tau1*s;'
'den=tau*s*(1+tau2*s);'
'Nsav=10000'
'Tfin=Nsav*Tsampl*30'
];
```

synthe_int_jit_sim_ctxt.sce

2.2.5 Mod_num blocks

- PFD_f - Phase/Frequency tristate Detector block
- VCO_f - Discrete Voltage Controlled Oscillator block
- PCLOCK_f - Integrated phase modulator event generator block
- NOISEBLK_f - Gaussian White Noise Generator block
- ZBUFT_f - Event discrete buffer block

2.3 Gaussian modulated fractional frequency synthesizer

2.3.1 Description

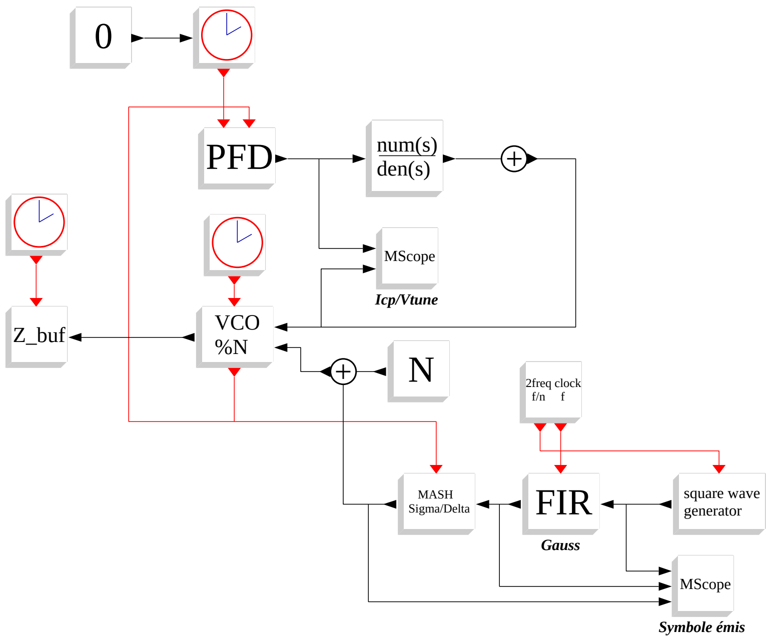
Add here a paragraph of the function description.

2.3.2 Simulation script(s)

```
//stacksize(1.8e8);
sim_name='synthe_sd_quick';
sim_path='synthe_sd_quick_sim';
load(MODNUM+'/simu/'+sim_path+'/'+sim_name+'_sim.cos');
exec(MODNUM+'/simu/'+sim_path+'/'+sim_name+'_ctxt.sce');
context=scs_m.props("context");execstr(context);
scs_m.props.tf=Tfin;

if exists('Info')==0 then Info=list(); end
if exists('%scicos_context')==0 then %scicos_context=struct(); end

Info=scicos_simulate(scs_m,Info,%scicos_context,"nw");
myvar=return_state_block(Info,"z_buf");
clear Info;
timer();
fc=2.49e9-2.5e6/2;
//step=plot_spectre2(myvar(1),Tsampl,Tacqui,fc,0.015e6,10,0);
step=1/(Nsav*Tsampl);
DSP=10*log10((1/(Nsav)*abs(fftshift(fft(myvar(1)(1:Nsav),-1))))^2);
lwin=0.45e6;
clf(10);
scf(10);
//plot2d((((Fd/step)-lwin):((Fd/step)+lwin))*step,DSP(Nsav/2+((Fd/step)-lwin:((Fd/step)+lwin
nNsav=2^17;
sm=pspect(nNsav/16,nNsav,'re',myvar(1)(1:Nsav)');
smsize=maxi(size(sm))/2;fr=(1:smsize)/(2*smsize);
nstep=1/(2*Tsampl*smsize);
Fc=Fd+F_cpf/2; Fdecal=50e6; Fmin=Fd-Fdecal; Fmax=Fd+F_cpf+Fdecal;
vec=int(Fmin/nstep:Fmax/nstep);

plot2d(vec*nstep,10*log10(1/(Nsav)*sm(vec))); xgrid();

t_cpu=timer();
Nbsym=int(Tfin/Ts);
var_tt="Center Freq.=%e, Sampling Freq.=%e\nNumb. of Sample=%d, Freq. step=%e\n";
printf(var_tt,fc,Fsampl,Nsav,step);
printf("t_cpu=%f, Numb. of simulated symbol=%d, symb/s=%f\n",t_cpu,Nbsym,Nbsym/t_cpu);
//plot_spectre2(myvar,Tsampl,Tacqui,N*F_cpf,0.5e6,11,1);
```

synthe_sd_quick.sce

2.3.3 Scicos diagram(s)



synthe_sd_quick_sim.cos

2.3.4 Context file(s)

```
scs_m.props.context=[
'lines(-1);'
'F_cpf = 20e6;'
'T_cpf = 1/F_cpf;'
'sig_ref=T_cpf*0.02/100'
'Fo = 2.045e9;'
'To = 1 / Fo;'
'wo=2*%pi * Fo;'
'kv = 100.5e6;'
'alpha=6.91e9;'
'beta2=0.15;'
'j_vco=0.0e6;'
'Fd=2.5e9;'
'N=int(Fd/F_cpf)'
'Nsampl = 4;'
'Tsampl = 1/(Fd*Nsampl);'
'Fsampl = 1 / Tsampl;'
'Icp = 5e-3;'
'Ileak=0.05e-6;'
'fn=F_cpf/280;'
'phi=%pi/4;'
'Ts=3/fn'
'Nech=12'
'BT=0.7'
'nb_coef=85'
'Te=Ts/Nech'
'Nbit=3'
'M=32'
'order_sd=3'
'[tau1,tau,tau2]=calcul_3eme_ordre(fn,phi,kv*2*%pi,Icp,N);'
```

```
's=poly(0,''s'');'
'num=1+tau1*s;'
'den=tau*s*(1+tau2*s);'
'Tacqui=20e-5'
'Nsav=2^21'
'Tfin=Tacqui+Nsav*Tsampl'
];
```

synthe_sd_quick_ctxt.sce

2.3.5 Mod_num blocks

- PFD_f - Phase/Frequency tristate Detector block
- VCO_f - Discrete Voltage Controlled Oscillator block
- PCLOCK_f - Integrated phase modulator event generator block
- MASHBLK_f - Delta-Sigma modulator block
- RIFGEN_f - Generic Finite Impulse Response filter block
- ZBUF_f - Discrete buffer block

Chapter 3

Simulations of communication systems

3.1 Output spectra of Delta-Sigma modulator

3.1.1 Description

This simulation script enables the analysis of spectrum of a MASH $\Delta - \Sigma$ modulator when the modulator works with a gaussian filtered input symbol.

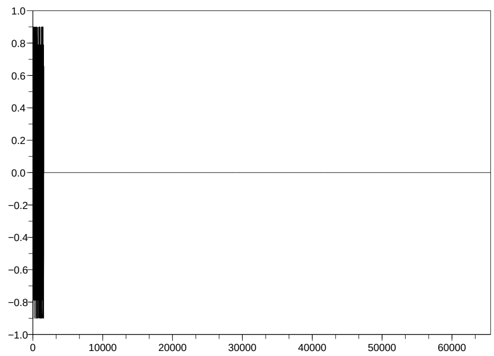
3.1.2 Simulation script(s)

```
//Define simulation name
sim_name='mash_gauss_sim';
sim_path='mash_gauss_sim';
//Load scicos diagram
load(MODNUM+'/simu/'+sim_path+'/'+sim_name+'.cos');
//Define context
exec(MODNUM+'/simu/'+sim_path+'/'+sim_name+'_ctxt.sce');
context=scs_m.props("context");execstr(context);
//Define Simulation end time
scs_m.props.tf=Tfin;
//Define other variables

for nb_r=1:3
  //substitue context variable to be sweep
  scs_m.props.context=subst_ctxt(scs_m,'order=','order='+string(nb_r));
  //Initialise Info and %scicos_context variable
  Info=list();
  if exists('%scicos_context')==0 then %scicos_context=struct(); end
  //Do simulation with scicos_simulate
  Info=scicos_simulate(scs_m,Info,%scicos_context,"nw");
  //Load result
  myvar=return_state_block(Info,"z_buf");
  if nb_r==1  then
    //draw time domain wave form of gaussian filtered input signal
    scf(10);
    myvart=myvar(1)(1:Nsav);
    plot2d(0:Nsav-1,myvart,rect=[0 -1 Nsav-1 1]);
    //compute fft of gaussian input signal
    DSP1=1/Nsav*abs(fftshift(fft(myvart,-1))^2);
    DSP1_log=10*log10(DSP1(Nsav/2:Nsav));
    //draw output sprectrum of gaussian filtered input signal
    scf(20);
    plot2d(0:Nsav/2,DSP1_log,rect=[0 min(DSP1_log) Nsav/2 max(DSP1_log)]);xgrid();
  end
  //draw time domain wave form of sigma-delta modulator output
```

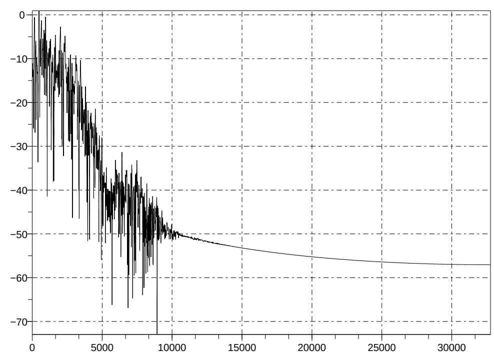
```
  scf(10+nb_r);
  plot2d(Nsav/4:Nsav/4+600,myvar(2)(Nsav/4:Nsav/4+600),rect=[Nsav/4 -2^nb_r Nsav/4+600 2^nb_
  //compute fft of sigma-delta modulator output
  myvart=myvar(2)(1:Nsav);
  DSP2=1/Nsav*abs(fftshift(fft(myvart,-1))^2);
  DSP2_log=10*log10(DSP2(Nsav/2:Nsav));
  //draw output sprectrum of sigma-delta modulator
  scf(20+nb_r);
  plot2d(0:Nsav/2,DSP2_log,rect=[0 min(DSP2_log) Nsav/2 max(DSP2_log)]);xgrid();
end
```

mash_gauss_sim.sce

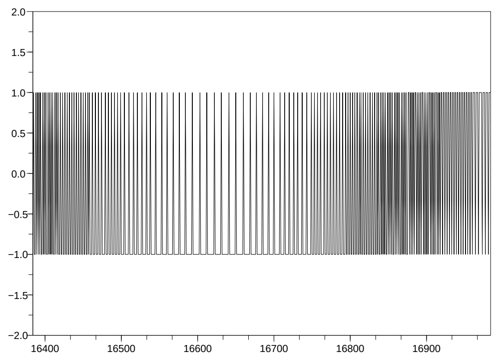
3.1.3 Scope Results



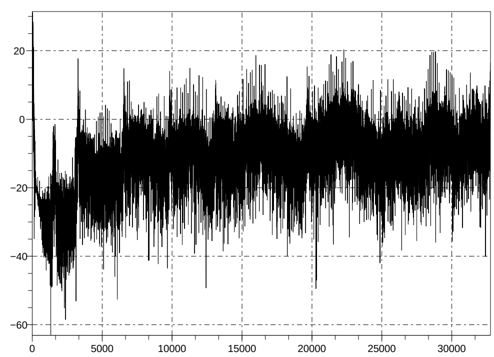
(a) Time domain wave form of gaussian filtered input signal



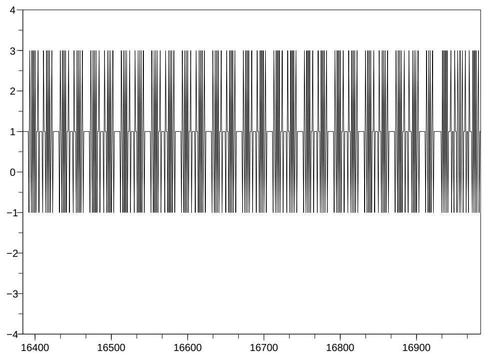
(b) Spectrum of gaussian filtered input signal



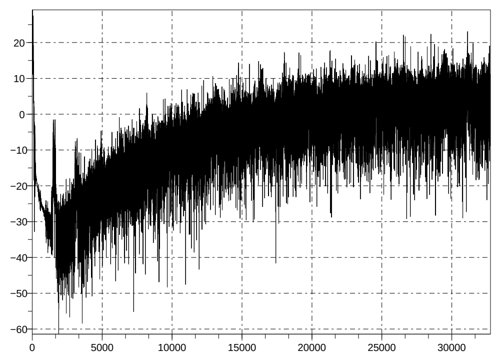
(c) Time domain output wave form of first order Σ-Δ modulator



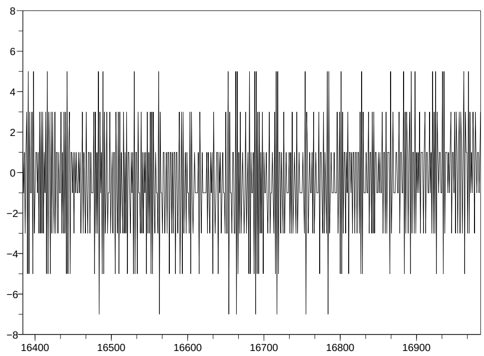
(d) Spectrum of output wave form of first order Σ-Δ modulator



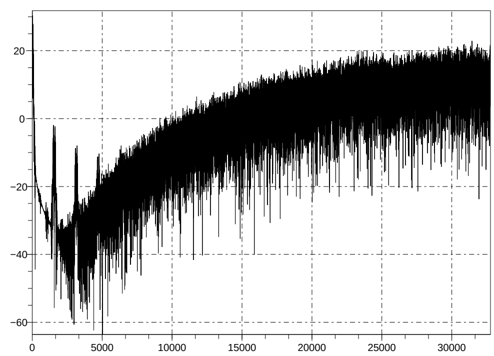
(e) Time domain output wave form of second order Σ-Δ modulator



(f) Spectrum of output wave form of second order Σ-Δ modulator

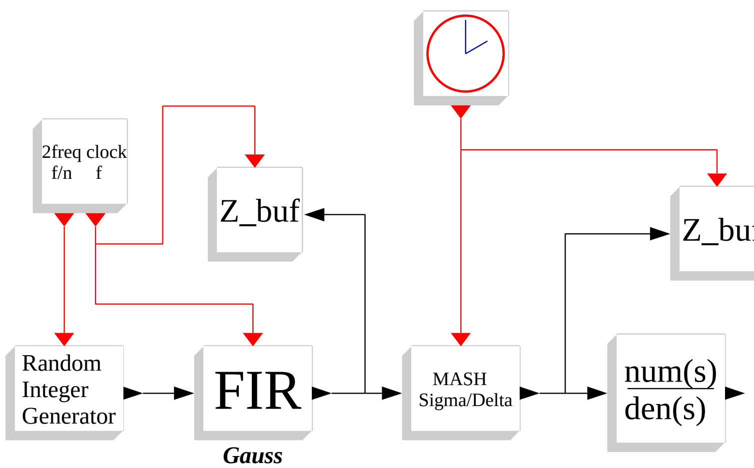


(g) Time domain output wave form of third order Σ-Δ modulator



(h) Spectrum of output wave form of third order Σ-Δ modulator

3.1.4 Scicos diagram(s)



mash_gauss_sim.cos

3.1.5 Context file(s)

```
scs_m.props.context=[
'M=1 '
'Te=1'
'Nech=12'
'order=3'
'Tsym=Te*Nech'
'Nsampl=500'
'Tsampl=Tsym/Nsampl'
'fc=0.03*(1/Tsampl)'
'eps=0.7'
'tau=1/(2*%pi*fc)'
'Nsav=2^16'
'Tfin=Nsav*Tsampl'
];
```

mash_gauss_sim_ctxt.sce

3.1.6 Mod_num blocks

- RIFGEN_f - Generic Finite Impulse Response filter block
- GENINT_f - Random Integer Generator block
- MASHBLK_f - Delta-Sigma modulator block
- ZBUF_f - Discrete buffer block

3.1.7 See Also

- mash_1er_ordre - Scattered diagram of first order Delta-Sigma modulator (Scicos Diagram)
- mash_2eme_ordre - Scattered diagramm of second order Delta-Sigma modulator (Scicos Diagram)
- mash_gauss - Integrated diagram of Delta-Sigma modulator (Scicos Diagram)

3.2 BER estimation of single user QPSK transmission

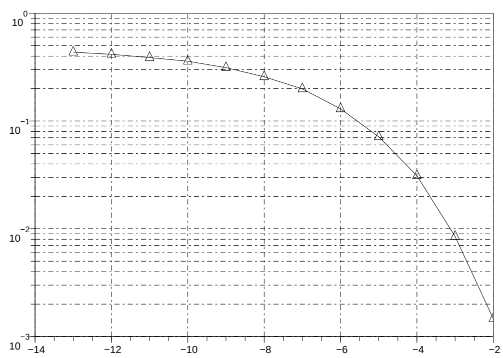
3.2.1 Simulation script(s)

```
//Define simulation name
sim_name='qpsk_teb_sim';
//Define simulation path
sim_path='qpsk_teb_sim';
//Load scicos diagram
load(MODNUM+'/simu/'+sim_path+'/'+sim_name+'.cos');
//Define context
exec(MODNUM+'/simu/'+sim_path+'/'+sim_name+'_ctxt.sce');
context=scs_m.props("context");execstr(context);
//Define Simulation end time
scs_m.props.tf=Tfin;
//Define other variables
//sig_log=-13:1:0;
sig_log=-13:1:-2;
Eb=cos(%pi*(1/(2^Nbit)));
sig=[];
for i=1:size(sig_log,2)
 sig(i)=Eb/10^(sig_log(i)/10);
end
nb_r=size(sig,1);
nb_error_sav=zeros(nb_r,1);

//Open Log file
u_log=mopen(MODNUM+'/simu/'+sim_path+'/'+sim_name+'.log','a');
//Set flag (for display saved information in scilab window)
flag=0;
//Save inital state of simulation in Log file
wlog_init(u_log,flag);
//Save context in log file
wlog_ctxt(u_log,flag);
//Save and display util variable
wlog_sv(u_log,tlist(['sig_log'],sig_log),flag);
l=1;
nbit_tot=0;
cpu_tot=0;
//while %t
while l<3
 nb_error=zeros(nb_r,1);
 //Loop for iterative simulation
 for j=1:nb_r
  //Save and display begin simulation date j
  wlog_bst(u_log,j,flag);
  //substitue context variable to be sweep
  scs_m.props.context=subst_ctxt(scs_m,'sigma','sigma='+string(sig(j)));
  //Initialise Info and %scicos_context variable
  //if exists('Info')==0 then Info=list(); end
  Info=list();
  if exists('%scicos_context')==0 then %scicos_context=struct(); end
  //Do simulation with scicos_simulate
  timer();
  Info=scicos_simulate(scs_m,Info,%scicos_context,"nw");
  cpu_time=timer();
  nb_bit=Nu*Nbit*(Nb_vec-nb_event);
  //Load result
  myvar=return_state_block(Info,"teb2");
  nb_error(j)=myvar(1)(1);
```

```
    cpu_tot=cpu_tot+cpu_time;
    nbit_tot=nbit_tot+nb_bit;
    bit_s=nbit_tot/cpu_tot;
    printf("cpu_tot=%f,\t nbit_tot=%d,\t bit/s=%f\n",cpu_tot,nbit_tot,bit_s);
    //Save and display util variable
    wlog_sv(u_log,tlist(['nb_error'],nb_error(j)),flag);
    //Save and Disp final simulation date j
    wlog_fst(u_log,j,flag);
  end
  //post-processed result
  //nb_bit=Nu*Nbit*(Nb_vec-nb_event);
  nb_error_sav=nb_error_sav+nb_error;
  teb=nb_error_sav/(l*nb_bit);
  plot_teb(49,sig_log,teb,"1");
  //Save and display post-processed vector(matrix)
  wlog_psv(u_log,tlist(['nb_bit';'nb_error_sav';'teb';'l'],nb_bit,nb_error_sav,teb,l,1));
  //
  l=l+1;
end
//Close final state of simulation in Log file
wlog_final(u_log,flag);
//close log file
mclose(u_log);
```

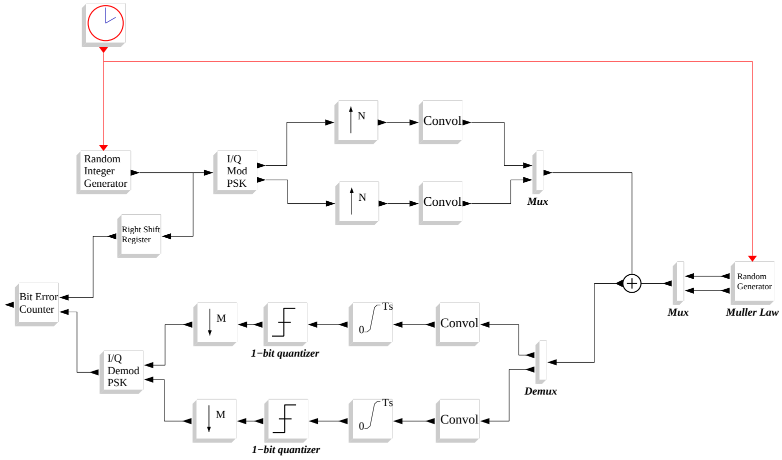
qpsk_teb_sim.sce

3.2.2 Scope Results



Scope results

3.2.3 Scicos diagram(s)



qpsk_teb_sim.cos

3.2.4 Context file(s)

```
scs_m.props.context=[
'Te=0.1;'
'Nu=16;'
'Nech=15;'
'N=Nu*Nech;'
'Nbit=2;'
'nb_coef=127;  '
'r=0.35; '
'fe=Nech; '
'gain_em=1; '
'gain_rec=1/Nech; '
'pulse=zeros(1,1); '
't_Ts=1/fe*(-nb_coef/2:nb_coef/2-1); '
'pulse = 4*r/%pi*(cos((1+r)*%pi*t_Ts) + (sin((1-r)*%pi*t_Ts)./(4*r*t_Ts)))./(1-(4*r*t_Ts).^
'Nb_vec=2^11; '
'nb_event=3;'
'Tfin=Te*Nb_vec'
'sigma=0.6;'
];
```

qpsk_teb_sim_ctxt.sce

3.2.5 Mod_num blocks

- CONVOLGEN_f - Convolution block
- INTSYMB_f - Discrete Symbol Integrator block
- MODPSK_f - M-ary Phase Shift Keying Modulator Block
- DEMODPSK_f - M-ary Phase Shift Keying demodulator Block
- OVERLAPRSR_f - Memory discrete shift register for multiplexed signal block
- NOISEBLK_f - Gaussian White Noise Generator block
- GENINT_f - Random Integer Generator block

- BITERROR_f - Binary error estimation block
- COMP_f - One bit Quantizer block
- UPSMPL_f - Up-sample block
- DOWNSMPL_f - Down-sample block

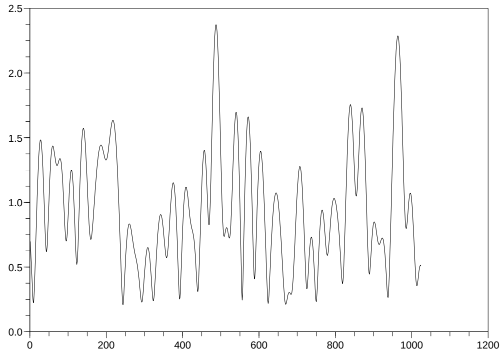
3.3 Rayleigh noise generator

3.3.1 Simulation script(s)

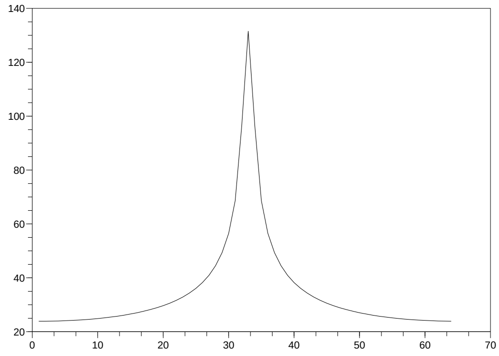
```
//Simulateur de processus de Rayleigh

//Le code original est un code Matlab qui a été fourni
//par J.P Cances (ENSIL) dans le cours de DEA en Traitement du
//signal numérique.

//Le code délivre un processus aléatoire de Rayleigh dont la Densité
//Spectrale de Puissance est déterminée par la vitesse d’un véhicule
//en mouvement (spectre de Jakos)

//RAZ des variables
//clear;

//polyfit est une fonction qui permet de déterminer les coefficients
//d’un pôlynôme donnée par un vecteur P(X). A l’origine c’est une fonction
//matlab qui a été traduite en scilab (trouvé sur le site des
//Newsgroup de scilab).

//getf(’./polyfit.sci’);

///////////////////////////////////
//Paramètres d’entrée de la simulation
////////////////////////////////////
fd = 0.02; //définition de la fréquence Doppler normalisée
fs = 1;    //definition de la fréquence d’échantillonnage
Ns=1024;   //définition du nombre d’échantillon de Rayleigh à produire

//Calcul du nombre d’échantillons aléaotoires en entrée
N = 8;
while (N)
 if (N<2*fd*Ns/fs)
   N = 2*N;
 else
   break;
 end
end

//Nombre de points de la fft inverse (pour lissage)
N_inv = ceil(N*fs/(2*fd));

//calcul de l’intervalle fréquentiel après la fft
delta_f = 2*fd/N;

//calcul de l’intervalle temporel des échantillons de sorties
delta_T_inv = 1/fs;


//Calcul de deux processus aléatoire gaussien (WGN)
I_input_time = rand(1,N,’normal’);
Q_input_time = rand(1,N,’normal’);

//Calcul des fft des deux processus
I_input_freq = fft(I_input_time,1);
Q_input_freq = fft(Q_input_time,1);
```

```
/////////////////////////////////////////////////////
//Calcul de la fonction de transfert du filtre Doppler
/////////////////////////////////////////////////////
//Composante continue
SEZ(1) = 1.5/(%pi*fd);

// 0 < f < fd
for j=2:N/2
  f(j) = (j-1)*delta_f;
  SEZ(j) = 1.5/(%pi*fd*sqrt(1-(f(j)/fd)^2));
  SEZ(N-j+2) = SEZ(j);
end

//utilisation de polyfit pour le calcul de la composante à f = fd
k=3;
p=polyfit( f(N/2-k:N/2), SEZ(N/2-k:N/2), k);

//approximation de la composante à partir des coef donné par polyfit
l=size(p,'c');
polyn=0;
x=poly(0,"x");
for i=1:l
polyn=polyn+p(i)*x^(i-1);
end
SEZ(N/2+1)=horner(polyn,f(N/2)+delta_f);

//mise en forme et affichage de la fonction de transfert
SEZ=matrix(SEZ,1,N);
xset("window",1);xset("wdim",300,200);xbasc(1);plot2d(SEZ);

//Réalisation du filtrage
I_output_freq = I_input_freq .* sqrt(SEZ);
Q_output_freq = Q_input_freq .* sqrt(SEZ);

//insertion de zéros pour le lissage
I_temp = [I_output_freq(1:N/2) zeros(1,N_inv-N) I_output_freq(N/2+1:N)];
Q_temp = [Q_output_freq(1:N/2) zeros(1,N_inv-N) Q_output_freq(N/2+1:N)];

//calcul des fft inverses des processus filtrés
I_output_time = fft(I_temp,-1);
Q_output_time = fft(Q_temp,-1);

//calcul du module de l'amplitude au carrée de la réponse temporelle
for j=1:N_inv
 r(j) = sqrt( (abs(I_output_time(j)))^2 + (abs(Q_output_time(j)))^2);
end

//normalisation de la réponse temporelle
rms = sqrt( mean (r.*r) );
r = r(1:Ns)/rms;

//affichage du processus de Rayleigh
xset("window",2);xset("wdim",300,200);xbasc(2);plot2d(r);
```

rayleigh_sim.sce



(a) Rayleigh noisy sample



(b) Impulse response of Doppler filter

3.3.2 Scope Results

3.3.3 See Also

- polyfit - polyfit mtlb equivalent function (Scilab Function)